Router SpaceWire
Análisis de diseño de un router para el espacio

Estudiante: Carlos Morales López
Profesor: Sergio López Buedo

Curso: 2012/2013
Fecha: 15/09/2013

Índice

Acrónimos

ABR	Available Bit Rate
AF	Assured Forwarding
BE	Best Effort
CAC	Call Admission Control
CBR	Constant Bit Rate
CLB	Component Load Balancing
CLS	Controlled Load Service
CORR	Carry-Over Round Robin
CPRR	Compensation Priority Round Robin
DCU	Digital Cryptographic Unit
DDRR	Dynamic Deficit Round Robin
DDS	Dynamic DiffServ Scheduling
DiffServ	Differentiated Services
DRR	Deficit Round Robin
EDAC	Error Detection And Correction
EF	Expedited Forwarding
FIFO	First-In, First-Out
FPGA	Field Programmable Gate Array
GPS	General Processor Sharing
GS	Guaranteed Service
HDL	Hardware Description Language
HDS	Hierarchical DiffServ Scheduling
HRR	Hierarchical Round Robin
IBP	Interrupted Bernoulli Process
ICU	Instrument Control Unit
IP	Internet Protocol
iSLIP	Iterative Round Robin with Slip
LUT	Look Up Table
MAC	Media Access Control address
MOQ	Multiple Output Queue
MT	Marco de Tiempo

OBC	On Board Computer
OQ	Output Queue
OSI	Open Systems Interconnection
PCU	Power Control Unit
PFQ	Packet Fair Queuing
PIFO	Push-In, First-Out
PIM	Parallel Iterative Matching
PQWRR	Priority Queuing - Weighted Round Robin
PWDRR	Priority Weighted Double Round Robin
PWRR	Priority Weighted Round Robin
QoS	Quality of Service
RAM	Random Access Memory
RIU	Remote Interface Unit
RR	Round Robin
SEU	Single Event Upset
SPI	Serial Peripheral Interface
TMR	Triple Modular Redundancy
VBR	Variable Bit Rate
VOQ	Virtual Output Queue
WDRR	Weighted Deficit Round Robin
WFQ	Weighted Fair Queuing
WRR	Weighted Round Robin

1. Introducción

1.1 Planteamiento del problema

El sector espacial es un ámbito tecnológico altamente ambiguo. Se busca constantemente realizar desarrollos muy innovadores, que representen avances significativos en campos tan importantes como las telecomunicaciones o la investigación. Sin embargo, las exigencias intrínsecas en este sector son muchas y diversas (alta redundancia, accesos específicos de depuración, robustez frente a errores provocados por la radiación…[JJACTEL]). Estas dificultades provocan cierto escepticismo a la hora de evolucionar determinados aspectos y componentes de los sistemas desarrollados para el espacio.

Si hablamos de la comunicación interna de un satélite, se observa una clara involución frente a los sistemas comerciales. Mientras que en tierra la mayoría de las telecomunicaciones usan protocolos orientados a paquetes, en el espacio aún se utilizan, muy ampliamente, protocolos casi obsoletos orientados a registros, como es el caso del Milbus 1553 [EC13].

En las nuevas misiones, incremento de la complejidad de los sistemas embarcados (cámaras de video con resoluciones de millones de píxeles "GAIA" [ES13], sistemas de encriptación de altísima complejidad, comunicaciones en tiempo real…) provoca un aumento casi exponencial en la tasa de datos entre instrumentos. Con el tiempo se han ido desarrollando diversos sistemas y protocolos para solucionar esta creciente problemática. Sin embargo, estas soluciones han sido parciales y ninguna ha conseguido aunar la problemática general. Esta situación está provocando un creciente e innecesario aumento en la complejidad de las comunicaciones internas del satélite así como de la cantidad de líneas físicas (comúnmente conocidas como hardness) necesaria para soportarlos. Todas estas razones nos muestran la, cada vez más palpable, necesidad de un protocolo serie, fácil de implementar, estandarizado, con bajo overhead y alto rendimiento así como un sistema de encaminamiento capaz de centralizar las comunicaciones internas de los satélites de nueva generación.

Con esta filosofía se creó el protocolo Spacewire[EC53], que ofrece una comunicación de datos en tiempo real entre cualquier instrumento como, por ejemplo, memorias, sensores o procesadores. A pesar de de la inicial reticencia de la industria aeroespacial, son cada vez más los ingenieros de sistemas que empiezan a ver el Spacewire como el "*protocolo IP*" para el espacio. Con un protocolo estandarizado que implemente o envuelva (ciertos sistemas son, a día de hoy, inevitablemente orientados a registros) todas o gran parte de las comunicaciones internas, se hace obvia la necesidad de un sistema de rutado automático dentro de un satélite.

Un router dentro de un satélite permitiría establecer una red interna de comunicaciones donde cada uno de los instrumentos (OBC, ICU, PCU, DCU, RIU…) sería entendido como un nodo capaz de conectarse a cualquier otro punto de la red.

1.2 Objetivos y planteamiento

Uno de los objetivos de este trabajo es comprobar la validez de los sistemas de enrutamiento comerciales para gestionar las comunicaciones internas de un satélite genérico. Para ello deberíamos conocer en profundidad el funcionamiento de estos sistemas así como las características que los diferencian.

La figura 1.1 nos muestra la estructura típica de un conmutador con soporte de paquetes IP.

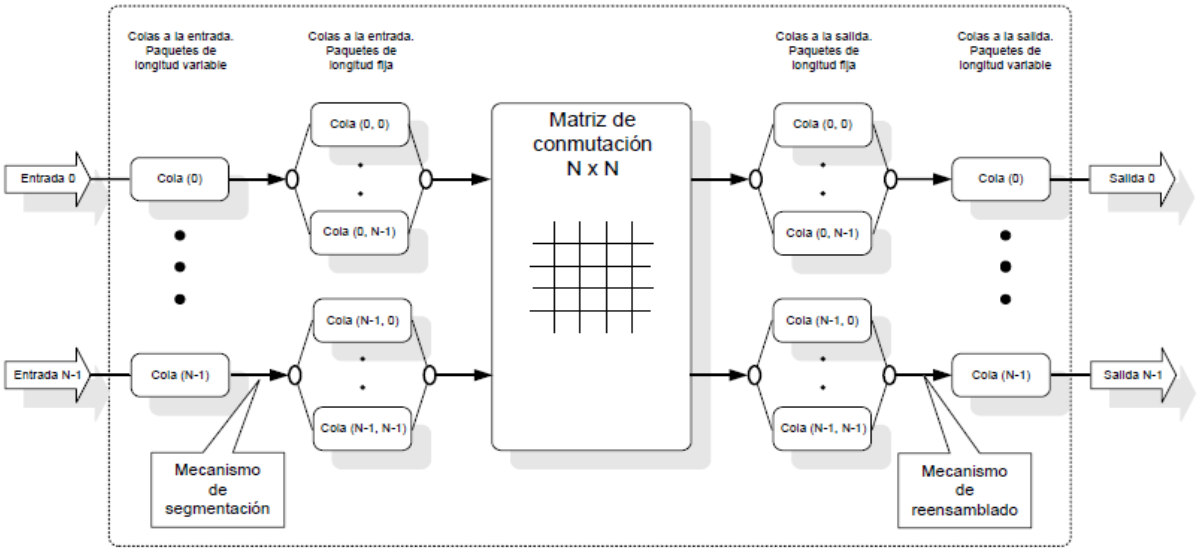

Figura 1.1: Arquitectura de un conmutador de alta velocidad con soporte para paquetes IP

Estos sistemas de encaminamiento evolucionan de forma paralela a las necesidades del tráfico de red que gestiona, el tráfico IP en el mundo comercial. Sus características se pueden categorizar según los principales requisitos de red: ancho de banda, retardo y tasa de pérdida de paquetes. La forma en que el sistema cumple y maximiza estos requisitos determina la estructura interna del router: número de puertos, colas de entrada y salida, colas virtuales [AN89].

También hay que tener en cuenta el tamaño de los paquetes que se van a gestionar y, más en concreto, la posibilidad de que el tamaño sea variable. Esta característica implicaría la necesidad de establecer un sistema de segmentación, que reduzca el paquete a segmentos de tamaño fijo, en los puertos de entrada [CYR+04][KP05] así como un sistema de reensamblado en los puertos de salida.

A estas características se les puede añadir la calidad de servicio (Quality of Service - QoS) que nos permita diferenciar de forma cualitativa los diferentes tráficos de datos que podemos encontrar en una red. Sin embargo, este último punto entraña una notable complejidad: dependiendo del tipo de arquitectura que tenga el conmutador, la implementación de QoS puede provocar situaciones en las que el tráfico de datos sea tratado de forma injusta, aumentando considerablemente el retardo de los flujos más

desfavorecidos. Esta reserva puede ser problemática dado que la asignación de determinados recursos puede reducir notablemente la eficiencia de la red provocando la disminución del ancho de banda, el aumento en el retardo y en la perdida de paquetes.

Todas estas características de los encaminadores comerciales son aplicables al sistema de rutado que se propone en este trabajo. Sin embargo es necesario acotarlas bajo las necesidades intrínsecas de la red de comunicaciones de un satélite.

El sector espacial es una parte de la industria tecnológica que tiene un alto grado de diferenciación. Por un lado, las necesidades del entorno nominal de funcionamiento de sus sistemas (baja o nula gravedad, temperaturas extremas, radiación cósmica) distan mucho de las encontradas en el mundo comercial. Por otro lado, la complejidad de los sistemas alojados, el carácter confidencial de muchos de ellos y la escasa cantidad (ver figura 1.2 [LES09**]**) dificulta la estandarización generalizada de estos sistemas.

Figura 1.2: lanzamientos espaciales 2009

Sin embargo, se pueden determinar las características más comunes de los satélites actuales y extrapolar así una arquitectura genérica que permita determinar los requisitos a satisfacer por el router.

Los satélites cuentan con un conjunto de subsistemas integrados para llevar a cabo todas sus funciones. Necesitan energía eléctrica, corregir su posición y movimiento, mantenerse en equilibrio, ser capaz de regular su temperatura, ser resistentes al medio en el que se encuentran, y poder comunicarse con la Tierra.

La estructura de un satélite se divide en dos conjuntos:

- Carga útil (Payload)
- Plataforma

La carga útil o payload está compuesta por todos los sistemas y unidades científicas que desarrollan la funcionalidad del satélite. Su importancia reside en que son las partes que el dueño del satélite puede modificar de acuerdo al propósito para el que será destinado el sistema. Por otro lado, la plataforma es el conjunto de subsistemas que dan el soporte necesario para que el satélite viva (control de vuelo, control de potencia (PCU), control de temperatura, control de spin….). En la figura 1.3 [SPO6] se puede observar el esquema interno de la plataforma del satélite SPOT-6. Se puede observar el complejo sistema de comunicaciones entre las diferentes unidades.

Ambos conjuntos de unidades sistemáticas son indiferentes desde el punto de vista de la comunicación interna del satélite. Con otro enfoque, se podría definir un satélite como un conjunto de "cajas negras" interconectadas que realizan una o múltiples funciones comunes. Estas cajas negras tienen implementados diferentes sistemas de comunicación que les permite compartir información vital.

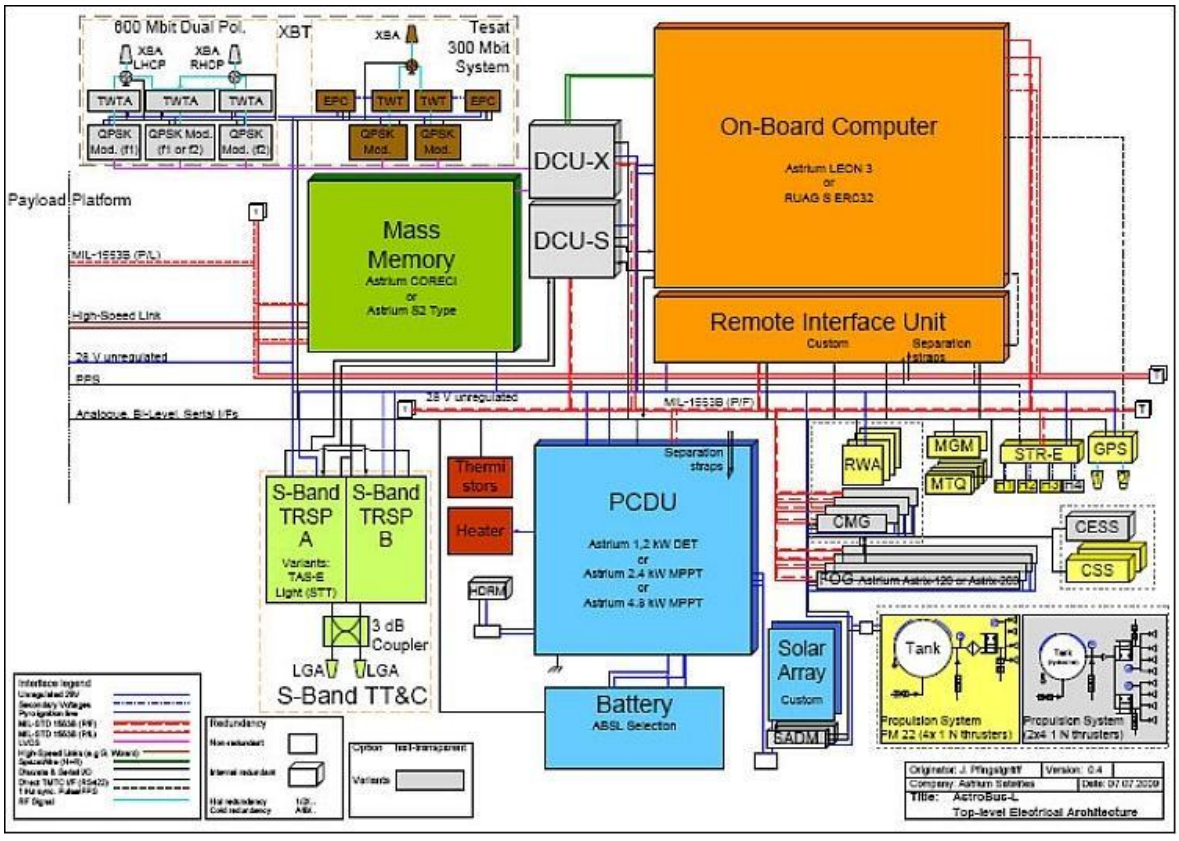

Figura 1.3: Estructura modular del satélite SPOT-6

Actualmente, las comunicaciones entre las diferentes cajas negras del satélite están restringidas a protocolos anticuados (por ejemplo MilBus1553) que, en algunos, casos limitan la funcionalidad de la propia caja. Muchos de estos protocolos están orientados a registros, produciendo la necesidad de redundar canales de comunicación por la falta de ancho de banda. Un protocolo orientado a paquetes como el SpaceWire podría dar soporte tanto a los instrumentos que necesiten comunicación de registros como aquellos que requieran tasas de transferencia mayores. Con un protocolo común, orientado a paquetes,

para todos los instrumentos del satélite se hace obvia la necesidad de un sistema de encaminamiento de flujos de datos, que centralice todas las comunicaciones del satélite. Observando la estructura planteada en las figuras y abstrayendo el funcionamiento particular de cada uno de los módulos que componen el satélite, se podría fácilmente crear una estructura de comunicaciones con un elemento central, un router con soporte para protocolo SpaceWire.

En posteriores capítulos se profundizará en el estudio de las comunicaciones internas del satélite ahondando especialmente en las necesidades de red que un supuesto router SpaceWire debiera satisfacer. Conocidas estas características, se puede generar un modelo representativo de alto nivel que describa la arquitectura más óptima para el router, con el fin de obtener sus prestaciones estimadas y poder presentar una descripción funcional detallada. En el presente trabajo se presenta un modelo descrito en lenguaje C que permitirá emular el funcionamiento de un sistema hardware real y determinar así la viabilidad del conmutador.

2. Arquitecturas de conmutación y algoritmos de planificación. Estado del arte

2.1 Definición

Un conmutador o switch es un dispositivo digital lógico de interconexión de redes de computadoras que opera en la capa de enlace de datos del modelo OSI (Open Systems Interconnection). Su función es interconectar dos o más segmentos de red, de manera similar a los puentes de red, pasando datos de un segmento a otro de acuerdo con la dirección MAC de destino de las tramas en la red.

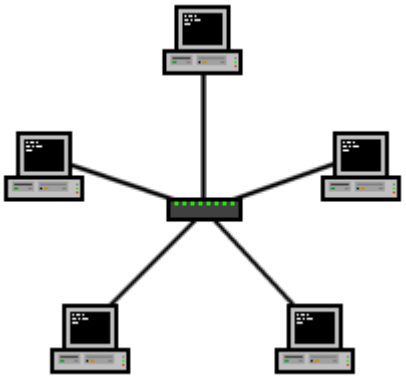

Figura 2.1: Un conmutador en el centro de una red en estrella.

Los conmutadores se utilizan cuando se desea conectar múltiples redes, fusionándolas en una sola. Al igual que los puentes, dado que funcionan como un filtro en la red, mejoran el rendimiento y la seguridad de las redes de área local.

Esta definición se puede ampliar asumiendo que los elementos conectados no necesariamente han de ser computadoras. Cualquier dispositivo que podamos identificar con una hard address y que tenga la capacidad de transmitir y recibir paquetes de datos es candidato a formar parte de una red. Así, podríamos establecer una red de datos, usando un router en el centro de la estrella y cada uno de los nodos sería un instrumento del satélite.

En la figura 2.2 [SEN5] se muestra la estructura interna del satélite SENTINAL-5. Se pueden apreciar los diferentes módulos como cajas negras independientes.

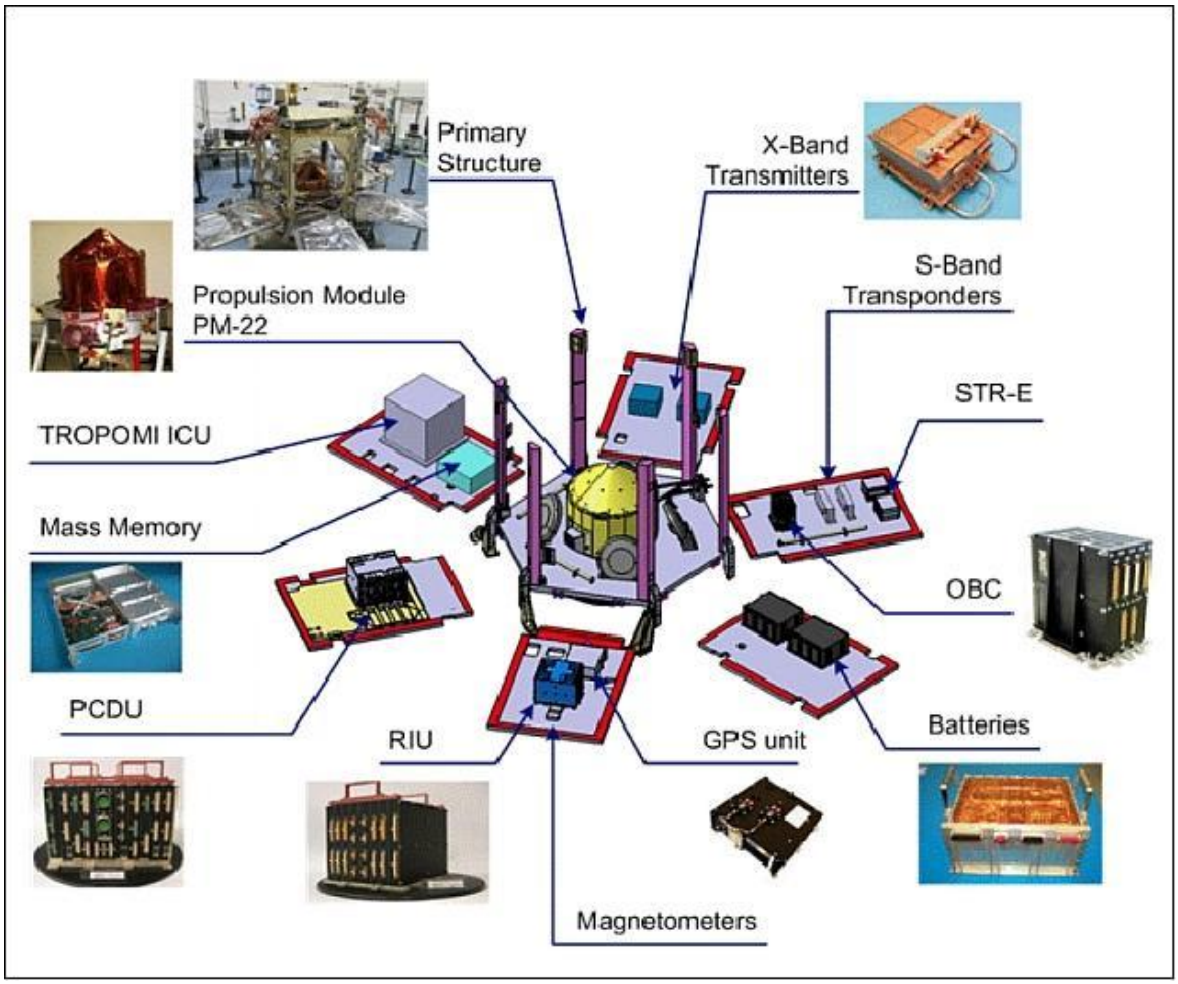

Figura 2.2: Estructura del satélite SENTINEL-5P

2.2 Arquitecturas de conmutación

Son muchas las características que determinan el funcionamiento de un conmutador: el modo de multiplexación de la información, la situación de los elementos de almacenamiento, la topología de la matriz de conmutación....

Todos estos aspectos son de gran importancia y la variación de cualquiera de ellos puede cambiar por completo el rendimiento del conmutador. Debido a la gran complejidad del estudio de estas características y la existencia de muchos y variados estudios sobre el tema: [Tob90][New92][RCG94][AM95][CLO01][Tse05], se centrará el estudio en el aspecto más determinante de todos, las técnicas de multiplexación.

2.2.1 Conmutadores por división de tiempo

Los conmutadores por división de tiempo se caracterizan por tener una única estructura o matriz de comunicación interna que comunica todos los puertos de entrada con todos los puertos de salida [IM01]. La estructura interna de conmutación puede implementarse mediante un bus, un anillo o una memoria.

Su principal ventaja es la facilidad de las transmisiones multicast pero la limitada escalabilidad de su estructura interna la convierten en una opción muy poco atractiva.

Una forma de mejorar las prestaciones de esta arquitectura es la llamada conmutación en medio compartido. La estructura interna se implementa en un medio con aceleración de tal forma que su velocidad de transmisión es N veces la velocidad de transmisión de los paquetes, siendo N el número de puertos de entrada. De este modo el throughput del sistema está únicamente limitado por la velocidad de transferencia de datos.

Figura 2.3: Conmutador con medio compartido.

2.2.2 Conmutadores por división de espacio

La principal diferencia con el anterior modelo es que los conmutadores por división de espacio proporcionan diversos caminos físicos entre los puertos de entrada y los de salida. Estas rutas funcionan de forma concurrente permitiendo la transmisión de varios paquetes por el conmutador de forma simultánea. El throughput teórico de este sistema es el resultado de multiplicar el ancho de banda de cada ruta por el número de paquetes que se puede transmitir de forma simultánea.

Dependiendo del número de rutas existentes entre dos puntos (puerto de entrada X – puerto de salida Y) se puede hacer una subdivisión y hablar de conmutadores de ruta única o conmutadores de ruta múltiple.

En cuanto a los conmutadores de ruta única existen tres ejemplos muy ilustrativos de esta clase de conmutadores: topología *crossbar* NxN, conmutador completamente interconectado y el conmutador Banyan.

El conmutador *crossbar* basa su funcionamiento en una matriz de conmutación NxN (siendo N el número de puertos) Figura 2.4. La matriz cuadrada está formada por N^2 elementos de conmutación Cada uno de esos elementos se activa de forma independiente al recibir la información que proporcionan los paquetes con el puerto de entrada y el puerto de salida.

Esta arquitectura posee una gran simplicidad de implementación y una alta modularidad del sistema, sin embargo el número de elementos de conmutación crece de forma exponencial, un problema de difícil solución [RCG94].

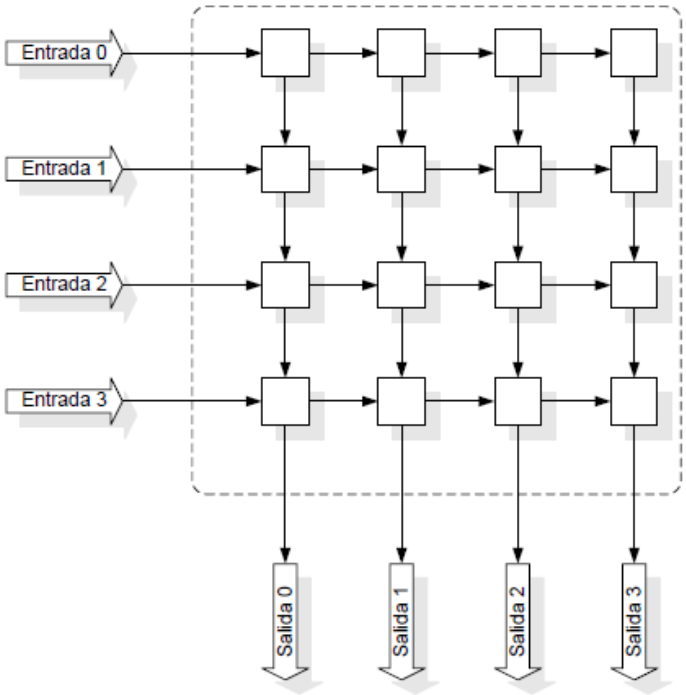

Figura 2.4: Arquitectura de un conmutador *crossbar* 4 x 4

Si interconectamos por hardware todos y cada uno de los caminos posibles hallaremos una topología de conmutación simple, rápida y no bloqueante, pero el coste en hardware es elevado.

El último ejemplo, es el conmutador *Banyan*. Este se caracteriza por estar compuesto por elementos de conmutación en forma de matriz 2 x 2, Figura 2.5. El número de elementos de conmutación es considerablemente más bajo que en el caso del *crossbar* o en caso del conmutador completamente interconectado y su crecimiento es exponencial con respecto al aumento de puertos.

Estos elementos se conectan en sucesivas etapas siendo ésta su principal desventaja, ya que en caso de que dos paquetes lleguen a la vez a un mismo elemento de conmutación, se producirá la pérdida de uno de ellos [Lea90].

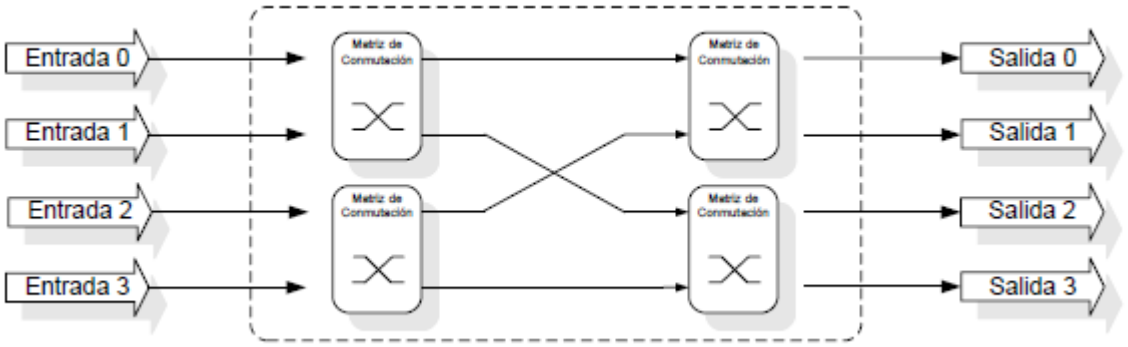

Figura 2.5: Arquitectura de un conmutador *Baynan* 4x4

2.3 Técnicas de almacenamiento

Cuando dos paquetes de datos distintos compiten por un mismo recurso esto recibe el nombre de colisión. En caso de que el elemento compartido que propicia la colisión sea un elemento de conmutación se dice que la matriz de conmutación es bloqueante. Si por el contrario, el recurso es un puerto de salida, se denomina contención de puertos de salida.

Ante esta indeseable situación el conmutador tiene dos opciones opuestas: descartar uno de los paquetes o almacenarlo.

Una primera solución es situar memorias independientes en los puertos de salida que permitan el almacenamiento de los paquetes en caso de colisión. Aplicando esta medida, relativamente simple, permite que el conmutador alcance un rendimiento óptimo entre el retardo de los paquetes y el throughput del sistema [HK88]. Sin embargo, el aumento del número de puertos y de la tasa de enlace hace que se incremente también la velocidad de las memorias y de la matriz de conmutación convirtiéndolas en el principal elemento limitador en la escalabilidad de los conmutadores con colas a la salida [SZ98] [MRS03]. Para paliar este problema, empiezan a aparecer modelos de conmutadores con colas en los puertos de entrada, con memorias internas…

2.3.1 Conmutadores con colas de entrada

Un conmutador con memorias internas independientes situadas en los puertos de entrada, es un conmutador que utiliza la técnica de almacenamiento con colas de entrada. Aunque hay muchas variantes, la más extensamente usada es aquella que usa la política FIFO (First-In, First-Out), esto es, los paquetes son atendidos según el orden de llegada. En la siguiente figura se muestra un esquema de la arquitectura.

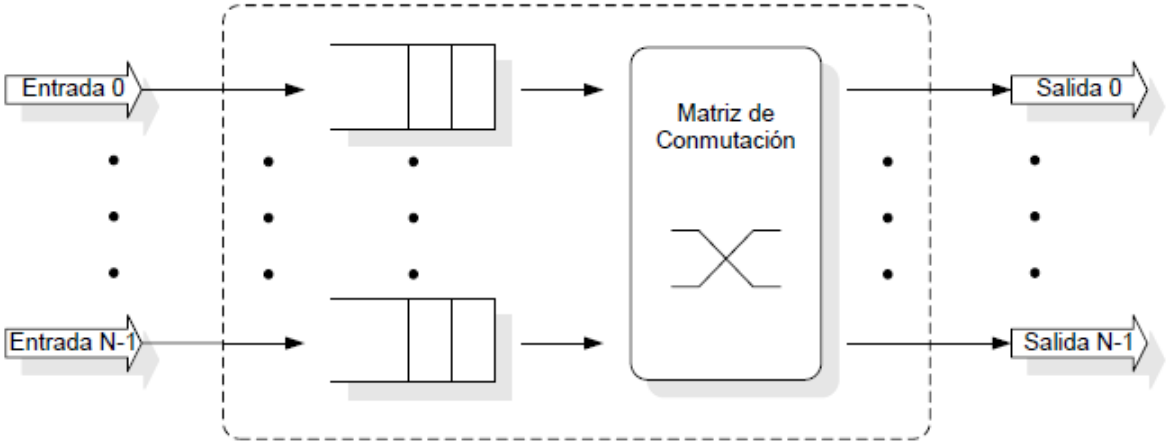

Figura 2.6: Arquitectura de un conmutador con colas a la entrada.

Esta arquitectura no requiere ningún "esfuerzo extra" por parte de la matriz de conmutación, es decir, no sería necesario ningún tipo de aceleración. El throughput del sistema es el resultado de la suma de los throughputs de las colas de entrada, lo cual la hace perfecta desde el punto de vista de la escalabilidad.

Sin embargo este modelo no está exento de problemas. La forma de funcionamiento de este conmutador consiste en almacenar los paquetes en las colas de entrada, generalmente usando la política FIFO, antes de la planificación y de la transmisión. Esto significa que solo conmutan aquellos paquetes que están en la primera posición de la cola forzando al os demás a esperar su turno. Este efecto se llama bloqueo de cabecera y limita el throughtput máximo a un 60%, estimado en [KHM87]. Otro problema añadido es la gran complejidad del algoritmo planificador, que por un lado debe solucionar los problemas de bloqueo de la cabecera y por otro evitar las colisiones en los puertos de salida.

Ante este problema surgen diversas soluciones: política de ventanas [KHM87], almacenamiento agrupado [TC94], expansión de los puertos de entrada [Mak98]…

Por su gran relevancia, cabe destacar una en particular: conmutación con colas virtuales de salida (Virtual Output Queue VOQ) [AN89]. La técnica consiste en desdoblar las colas de entrada, de tal manera, que en cada puerto de entrada haya tantas colas (se asume FIFO) como puertos de salida haya. Así se soluciona el problema del bloqueo de cabecera.

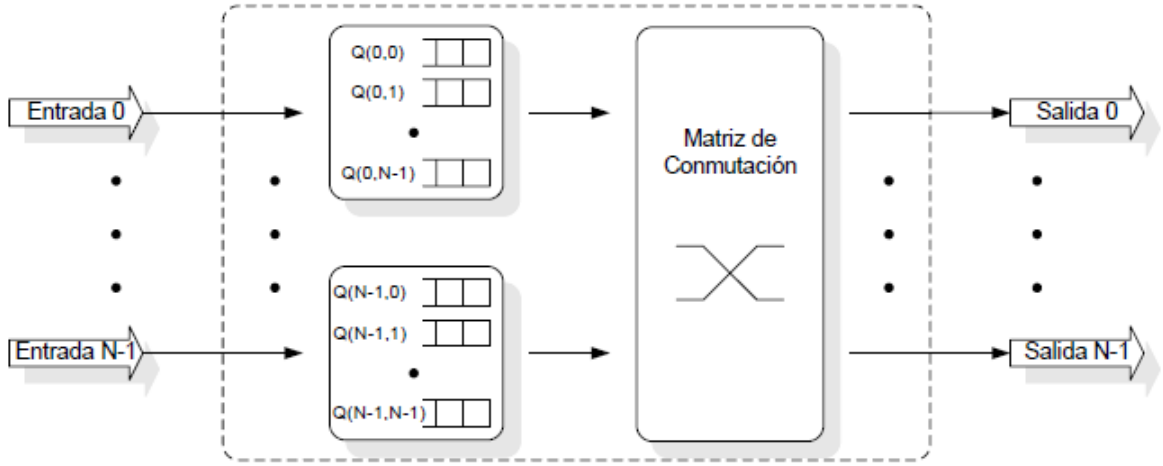

Figura 2.7: Arquitectura de un conmutador con colas virtuales de salida.

2.3.2 Conmutadores con colas de salida

Al situar las colas en el otro extremo del conmutador se obtiene un sistema completamente diferente. En este caso, el planificador debe atender los paquetes de entrada según lleguen situando estos en las colas que están en los puertos de salida, de tal forma que se evita cualquier tipo de colisión.

El principal problema de este modelo es la necesidad de aceleración de la memoria que aloja las colas de salida. En el peor de los casos, para un conmutador de NxN puertos, la matriz de conmutación recibirá N paquetes al mismo tiempo obligándola a tener una velocidad N veces mayor que la velocidad del enlace externo. Aunque en términos de retardo y de throughput es una técnica óptima [HK88] su escalabilidad es limitada.

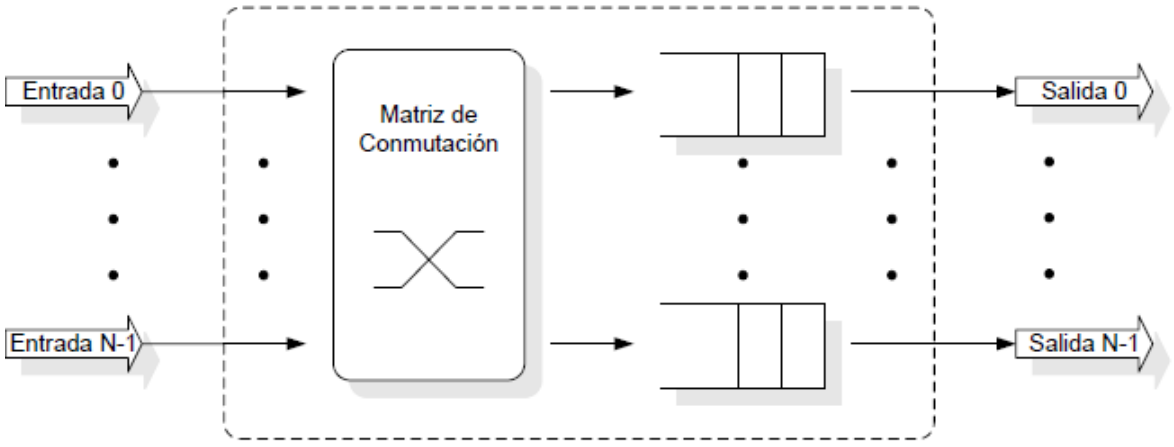

Figura 2.8: Arquitectura de un conmutador con colas de salida.

2.3.3 Otras técnicas de conmutación

Una solución intermedia seria situar una memoria compartida a la que todos los recursos tuvieran acceso, de tal forma que mientras los puertos de entrada sitúan los paquetes entrantes en la memoria intermedia, los puertos de salida cogen los paquetes que se les ha asignado. Como se puede observar en el experimento Prelude [DCS88] este método maximiza el uso de la memoria y reduce al mínimo la posibilidad de colisión. El rendimiento (desde el punto de vista del retardo y del throughput del sistema) es equivalente al rendimiento de un conmutador con colas de salida. El gran problema de esta topología es la altísima velocidad de la memoria que debe permitir el acceso simultáneo de todos los puertos de entrada y todos los puertos de salida.

Una alternativa es la modificación de los modelos crossbar y Banyan. Situando memorias distribuidas delante de los elementos de conmutación de estos elementos se evita las posibles colisiones en caso de dos intentos de acceso simultáneo a un mismo recurso.

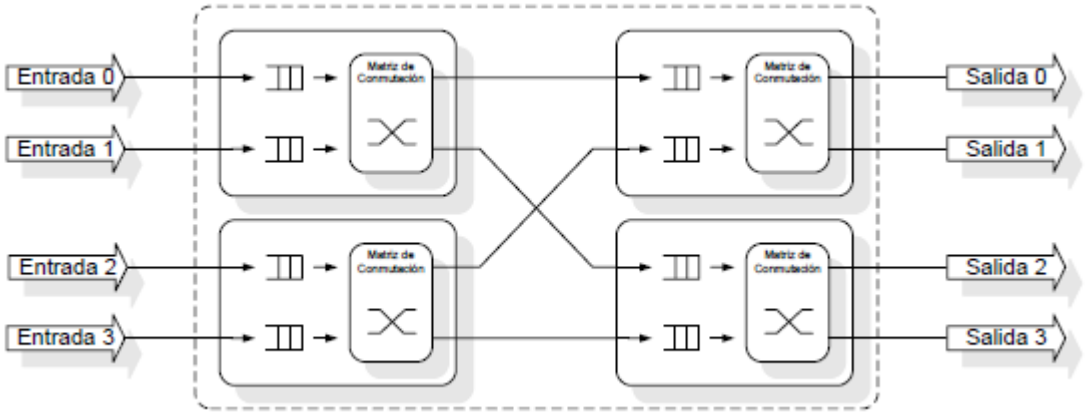

Figura 2.9: Arquitectura *Banyan* modificada

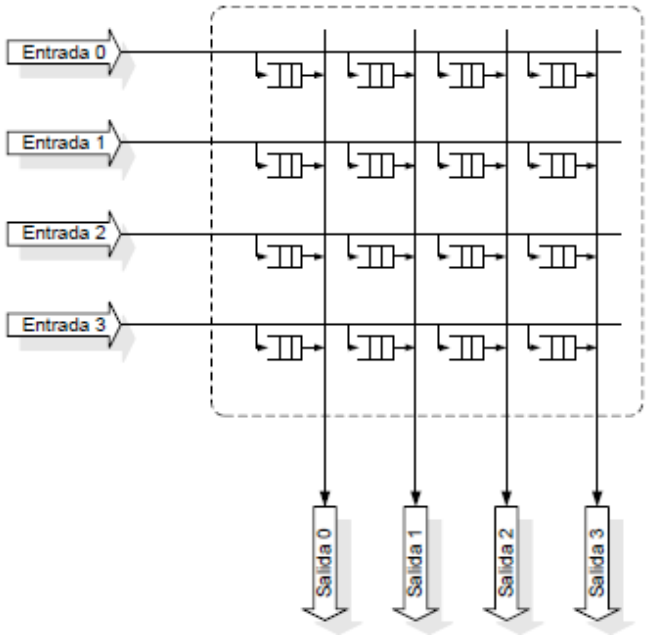

Figura 2.10: Arquitectura *crossbar* modificada

Un modelo especialmente interesante es el propuesto por Nabeshima [Nab00]] en el año 2000. El modelo presenta un conmutador con colas virtuales de salida y colas en los elementos de cruce de la matriz de conmutación *crossbar*, lo cual reduce de forma significativa la memoria interna de la matriz de conmutación. El algoritmo de planificación elige el paquete de la cabecera que lleve más tiempo encolado, de entre todos los almacenados en las en los puertos de entrada y en los elementos de cruce de la matriz eliminando así el bloqueo de cabecera y reduciendo el retardo medio observado en el conmutador de con colas de entrada.

El conmutador propuesto en [YC01], representa una clara evolución del anterior con la ventaja de soportar paquetes de longitud variable sin necesidad de segmentador ni reensamblado. Se demuestra que esta arquitectura tiene un retardo menor que una con colas virtuales de salida y política de planificación iSLIP (iterative round robin with SLIP) [McK95]] para la conmutación de paquetes de longitud fija.

2.4 Gestión de tráfico

En los anteriores capítulos se ha realizado un profundo análisis de las arquitecturas de conmutación y de las técnicas de almacenamiento más significativas. Hasta ahora se ha asumido la política FIFO como la única posible en el manejo de los paquetes almacenados en las colas.

Sin embargo la gestión de estos paquetes puede mejorarse con diversas técnicas que permitan, entre otras cosas, dar soporte a paquetes de longitud variable de forma efectiva o proporcionar calidad de servicio (QoS) [Wan01] [CG02]. Para lograr estos objetivos se requiere la aplicación de un algoritmo parametrizado. En los capítulos siguientes se expondrán diversas técnicas para gestionar eficientemente los paquetes almacenados.

2.4.1 Tecnologías de la red

Existen dos enfoques distintos a la hora de abordar la problemática presentada. Si el enfoque está más orientado al propio flujo de datos se puede hablar de Servicios Integrados. En cambio, si se opta por centrar la atención en cada paquete por separado (origen, destino, tamaño, prioridad…) estaremos ante los llamados Servicios Diferenciados.

a. Servicios Integrados

Antes de definir el concepto de servicio integrado se hace necesario definir la idea en que se basa este concepto: el flujo de datos. Se comprende por flujo de datos al caudal de información simple y distinguible que proviene de un mismo punto y que mantiene y requiere mantener las diferentes características intrínsecas que lo definen (QoS).

Para lograr que la comunicación mantenga estas características de forma constante, los conmutadores deben controlar de forma directa tanto el ancho de banda como la memoria. Por ello, se puede deducir que la gestión de memoria, el control de admisión, la planificación del tráfico y la reserva de recursos son aspectos clave en los Servicios Integrados.

No siempre se puede tener el control absoluto de todos los aspectos relacionados con la conexión y, en muchos casos, si se puede no es eficiente. Por lo tanto, dentro de las conexiones con Servicios Integrados, se pueden distinguir dos clases bien diferenciadas: las que tienen un retardo fiable y acotado, llamadas servicios garantizados (Guaranteed Services – GS) y las que tienen un control del tráfico a costa de tener un retardo variable y soportar pequeñas perdidas de paquetes. Estas últimas son las llamadas conexiones con control de carga (Controled Load Service – CLS).

b. Servicios diferenciados

Los Servicios Diferenciados presentan una visión completamente opuesta a los Servicios Integrados. Su visión del tráfico se centra en el paquete individual, "obviando" el conjunto. Cada uno de los paquetes tiene reservados una pequeña cantidad de bits en su cabecera con información que permite al conmutador clasificarlo según sus necesidades. De esta forma se logra agrupar los paquetes de características y necesidades similares facilitando su almacenamiento y simplificando su procesamiento. Dado que cada paquete se procesa de manera individual en este tipo de servicios, no existe la reserva de recursos [BBC+98].

2.4.2 Calidad de servicio

Uno de los aspectos más importantes en una red de conmutación es poder garantizar sus características (mantener una baja tasa de pérdida de datos o garantizar un retardo acotado) de transferencia de datos en cualquier situación [Wan01] [CG02].

Con el fin de lograr la máxima flexibilidad posible, la red de conmutación dispone de diversos mecanismos y políticas: transferencia con prioridades, planificación de paquetes, política de control y conformación de flujos de datos o gestión de recursos de memoria.

2.4.3 Planificación de paquetes

Una red de paquetes permite que cada punto de enlace comparta los recursos generales de la red (memoria, ancho de banda) con el resto de enlaces. Esta situación provoca efectos indeseados como la contención en los puertos de salida. Para solucionar este problema, se necesita un planificador que establezca el orden en que se transfieren los paquetes. Se hace necesaria la aplicación de un algoritmo de planificación que sea capaz de asignar prioridades a cada uno de los paquetes y que solvente la problemática de la contención de los puertos de salida.

Son varias y diversas las características que definen a un algoritmo de planificación. Una de las más importantes es el tiempo que utiliza para realizar las operaciones de selección y encaminamiento de paquetes. El objetivo para todo algoritmo es reducir este tiempo, en la medida de lo posible. A la vez es deseable, que la variación de su comportamiento en el tiempo y su complejidad sean lo más bajas posibles. Otra característica a tener en cuenta es que, independientemente de su cantidad o de la QoS asociada, debe proporcionar un tratamiento justo a cada uno de los diferentes flujos de datos. Asumiendo la variabilidad del comportamiento en función de la carga de tráfico, el algoritmo debería garantizar que el retraso asociado a la clase de tráfico más desfavorecida no fuera excesivo.

Desgraciadamente, algunas de estas características pueden llegar a presentarse como contrarias. La simplicidad y el compromiso temporal entran habitualmente en conflicto con la justicia. Los planificadores justos a corto plazo y con un retardo acotado son a su vez, muy complejos en el tiempo y difíciles de implementar. Los esquemas de mayor simplicidad suelen presentar una planificación poco justa o ser incapaces de asegurar un retardo bajo.

Como ejemplo de esta contradicción se pueden citar lo algoritmos de planificación basados en sellos de tiempo. Estos algoritmos usan la generación de un reloj virtual con el fin de emular el algoritmo Generalized Processor Sharing (GPS) [PG93] y poder así atender los flujos de datos utilizando como unidad mínima los paquetes (Packet Fair Queuing – PFQ) en lugar de flujos infinitesimales. El Weigthed Fair Queuing (WFQ) [DKS89][PG93] es un ejemplo de este tipo de algoritmos de planificación. Este algoritmo se caracteriza por un límite de retardo bajo y su buena justicia, sin embargo su complejidad temporal es alta O(N) (siendo N el número de flujos activos). Existen variantes de este algoritmo como Virtual Clock [Zha90] o Wort Case Fair Weighted Fair Queuing (WF^2Q) que, usando diferentes métodos para calcular el sello de tiempo, mejoran su complejidad hasta O(logN), sin embargo sigue siendo demasiado elevada.

En el otro extremo, existen algoritmos basados en la filosofía Round Robin (RR) que garantizan una complejidad temporal baja O(1) pero que presentan tienden a mostrar injusticias y a generar ráfagas de tráfico en un corto período de tiempo. Carry-Over Round Robin (CORR) [SMT98] o Deficit Round Robin (DRR) [SV96] son dos ejemplos de esta filosofía.

2.4.4 Políticas de planificación con calidad de servicio

Antes de entrar en el análisis de las posibles arquitecturas para el conmutador se hace necesario explicar una seria de términos relativos a la caracterización y cuantificación de las propiedades de estos dispositivos. La calidad de servicio puede expresarse con varios parámetros como el throughput, el retardo, el ancho de banda, la probabilidad de pérdida de paquetes o la variación del retardo (jitter).

a. Throughput

Entendemos por throughput de un sistema de comunicaciones como la cantidad media de información que un canal es capaz de enviar.

Este término es de vital importancia cuando se habla de las arquitecturas de conmutación con colas a la entrada y todas sus topologías. Los puertos de entrada y de salida de un conmutador son interconectados gracias al algoritmo de planificación y a la técnica de almacenamiento interno y por ello son estos dos aspectos clave los que determinan el throughput del conmutador. Para alcanzar el máximo throughput se usa la técnica de almacenamiento con colas de salida combinada con una política de planificación conservadora [HK88][OMK+89].

b. Control de ancho de banda

Actualmente, las arquitecturas de conmutadores más comunes usan técnicas de almacenamiento con colas virtuales de salida. De este modo se elude la problemática del bloqueo de cabecera y la necesidad de aceleración interna, problemas comunes en los conmutadores con colas de entrada. Sin embargo, estas ventajas tienen un sobrecoste asociado, y este es la necesidad de sincronizar los puertos de entrada con los puertos de salida para maximizar el throughput [MMA+99] y reducir la complejidad del planificador [GKS05].

Otro problema típico de los conmutadores es el tratamiento de paquetes de longitud variable. Por una parte, la conmutación directa de paquetes, puede parecer a priori, la solución más directa, al no necesitar mecanismos adicionales de segmentación ni reensambado (en la conmutación segmentada los paquetes de longitud variable se parten en células más pequeñas). En la realidad, esta técnica eleva la complejidad de la política de planificación y disminuye el throughput [MS01] dado que introduce overhead.

Por ello, la técnica más habitual de manejar paquetes de longitud variable es la segmentación de estos en células más pequeñas. Esta segmentación se lleva a cabo en los puertos de entrada y, después de ser transmitido, se reensamblan en los puertos de salida. Durante todo el proceso de transmisión se debe garantizar un retardo constante para todas las células que componen el paquete, dado que si una sola de ellas se bloquea en los puertos de entrada las restantes tendrán que esperar en el puerto de salida hasta que ésta se transmita [MS01][DY02]. Además, la perdida de una célula implica la pérdida total del paquete. Este aspecto incrementa la complejidad del algoritmo. No hay que olvidar la importancia de la elección del tamaño de la célula [CYR+04] [KP05].

Estas características del algoritmo determinan, en gran medida, el ancho de banda y el retardo del algoritmo resultante [MBG+01][GKS05].

En el caso de elegir un tamaño de paquete fijo, el algoritmo Weighted Round Robin (WRR) [KSC91] es una de las mejores opciones. Este algoritmo asigna un peso a cada paquete contenido y lo sitúa en una cola de entrada que contiene paquetes del mismo peso. El planificador hace una consulta Round Robin por todas las colas pero en vez de enviar un solo paquete envía tantos como el peso asignado indique. Lo interesante de este algoritmo es la flexibilidad que permite pues en cada caso la asignación de pesos se puede ajustar al ancho de banda disponible en ese momento.

Pero la filosofía Round Robin no solo es útil para los paquetes de longitud fija. El algoritmo Deficit Round Robin (DRR) es capaz de lidiar con paquetes de longitud variable de forma eficiente. En realidad este algoritmo no es más que una evolución del Round Robin clásico. Básicamente, cada cola de entrada tiene asignado un porcentaje del servicio llamado Quantum. Cuando el Round Robin asigna recursos a una determinada cola comprueba si el servicio asignado es suficiente para satisfacer el paquete acumulado, en caso de no serlo, se le incrementa al Quantum, se almacena y se espera hasta la siguiente iteración. Por cada paquete se realizarán tantas iteraciones como la relación Quantum/tamaño lo indique. La granularidad de este algoritmo provoca que no sea capaz de gestionar eficientemente paquetes de pequeño o mediano tamaño combinados con un alto throughput. Este aspecto es solucionado por el Dynamic Deficit Round Robin (DDRR) [YNO+02] .

Para el caso de combinar paquetes de longitud variable con diferentes pesos en las colas de entrada, la solución más óptima es el Weighted déficit Round Robin(WDRR) [WL02] Esta última solución presenta problemas de reparto del ancho de banda, no siendo capaz de soportar diferentes calidades de servicio.

c. Control de retardo. Prioridad de los paquetes

A principio de la década de los '90 Kalmanek et al. [KKK90] propusieron un planificador llamado Round Robin Jerárquico (Hierarchical Round Robin – HRR) para paquetes de longitud fija. Este algoritmo se basa en la segmentación del tiempo en unidades llamadas marcos (de longitud constante) y en subunidades más pequeñas llamadas ranuras. A cada paquete se le asigna un marco o una ranura cuando entra en el puerto de entrada. Este algoritmo, así como otros similares (véase el algoritmo Stop and Go [Gol91]) presentan dificultades en el acoplamiento entre el retardo y la granularidad del ancho de banda asignado [CG02]. Consecuentemente, se requieren marcos de tiempo grandes para asignar flujos con bajas exigencias de ancho de banda y marcos de de tiempo breves para proporcionar un retardo reducido.

Afortunadamente, los marcos de tiempo no son la única herramienta disponible para controlar los retardos. La prioridad de un paquete determina la importancia del flujo de datos al que pertenece. Consecuentemente, un flujo de datos o un paquete perteneciente a un flujo de datos con mayor prioridad enviará sus datos antes que otro con menor prioridad. Por ello, los paquetes entrantes deben ser marcados con una determinada prioridad que determinara el tiempo que estará almacenado. Adicionalmente, este procedimiento simplifica el trabajo del planificador, ya que, en un momento dado, solo compiten los paquetes con la misma prioridad.

Siguiendo esta filosofía, se propone una arquitectura de planificación consistente en varias colas de entrada [KKL00]. Un primer planificador estático selecciona una cola para crear una conexión basándose en la prioridad. A posteriori, un planificador dinámico Parallel Iterative Matching PIM [AOS+93] selecciona el puerto de salida con el que completar la conexión.

Al igual que se desarrollaron algoritmos de planificación basados en el pensamiento Round Robin, en el caso de la planificación ideal GPS también ha habido aportaciones. Es de destacar el Worst Case Fair Weighted Fair Queuing (WF^2Q+) [BZ97] que proporciona un límite de retardo bajo combinado con un índice alto de justicia.

Una opción para combinar paquetes de longitud variable con calidad de servicio, es emplear el algoritmo (ya descrito) DDRR. Una primera opción es la asignación de un token a cada paquete en base a su peso y prioridad, el DDRR seleccionara entre la lista de tokens [WWL05].

Otra opción algo más depurada se realiza en dos etapas: en la primera se filtra atendiendo al origen del paquete y en la segunda etapa se planifica según su peso [WWM+06].

En caso de necesitar paquetes de longitud variable, se puede usar el algoritmo WRR combinado con un algoritmo de prioridad estática para ofrecer calidad de servicio: Priority Queuing Weighted Round Robin (PQWRR) [MMW01].

El uso de este algoritmo se pormenoriza en [ADF04]. Resumiendo, los paquetes entrantes son clasificados en tres clases de servicio distintas, atendiendo a su prioridad: EF (Expedited Forwarding), con ancho de banda ilimitado y máxima prioridad; AF (Assured Forwarding) que serán ponderadas por el WRR pero su transmisión está asegurada y con mayor prioridad que la última clase; BE (Best Effort Forwarding), que no se pondera con WRR y no tiene ancho de banda asegurado.

Otra revisión de este mismo algoritmo [ZH07], el WRR pondera también las clases AF y BE con el fin de asegurar un mínimo de ancho de banda a todas las clases.

La figura 2.11 escenifica un esquema del funcionamiento de la última versión del algoritmo Priority Weighted Round Robin (PWRR).

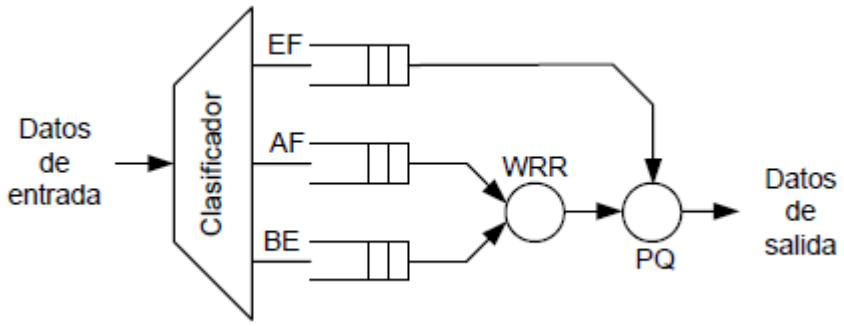

Figura 2.11 Panificador jerárquico PWRR

2.5 Arquitecturas de conmutación con QoS

A pesar de todas las arquitecturas propuestas hasta ahora, son muy pocas las soluciones enfocadas a arquitecturas escalables. La mayoría de las arquitecturas nuevas están orientadas a dar soporte a diferentes calidades de servicio usando colas de salida o memoria compartida, características que de por si limitan la escalabilidad y dificultan su implementación física.

2.5.1 Colas virtuales de salida

Se puede considerar [SPS99] como una primera implementación de un modelo con colas virtuales de salida. Esta propuesta presenta un algoritmo capaz de configurar prioridades con el fin de dar soporte a diferentes tipos de tráfico: tasa constante (Constant Bit Rate CBR), tasa variable (Variable Bit Rate VBR) y sin restricción (Available Bit Rate ABR) Sin embargo, está limitado a solo dos prioridades distintas.

Buscando un enfoque distinto, Chiusi y Francini [CF99] presentan una arquitectura jerárquica para resolver la calidad de servicio. En pocas palabras: se combina tres algoritmos de planificación diferentes WF^2Q, WRR y RR (eligiendo el más óptimo en cada caso en función del tráfico) y una segunda etapa de prioridad estricta. El resultado es un algoritmo con altísima flexibilidad, y con un coste de implementación considerable.

Otro planteamiento que proporciona calidad de servicio en una arquitectura con colas virtuales en la salida es el propuesto en [TEA+03]. En este caso, el algoritmo de planificación se divide en dos. Por una parte, un planificador se encarga de diferenciar las colas virtuales en los puertos de entrada y salida. Por otra parte, un planificador situado en la matriz de conmutación es el encargado de atender a las distintas prioridades. Interesante propuesta que aumenta la escalabilidad del sistema sin empeorar su rendimiento.

En [YLZ03] se presenta un algoritmo de control dinámico de ancho de banda (Dynamic DiffServ Scheduling DDS). Sobre éste último, se apoya el algoritmo de planificación Hierarchical DiffServ Scheduling HDS) [YWL+04]. Este algoritmo consta de dos partes. Por un lado, usando las diferentes prioridades, se seleccionan los paquetes en las colas virtuales de salida. Después de eso, un algoritmo combina los puertos de entrada previamente seleccionados con los puertos de salida con el único fin de maximizar el throughput final. Este algoritmo establece un ancho de banda mínimo para EF y AF, asegura un reparto justo del ancho de banda para BE con una complejidad inferior a la del DDS.

Existe un problema subyacente en las arquitecturas con colas virtuales de salida. Por definición, la matriz de conmutación de una arquitectura con colas virtuales de salida, tiene una capacidad limitada debido al problema inherente de la contención en los puertos de salida. Ocasionalmente puede ocurrir que un paquete no sea transmitido puntalmente entre los puertos, consecuentemente este paquete pierde la oportunidad de recibir servicio en los puertos de salida. Esta situación puede llegar a provocar la pérdida de algún parámetro de calidad de servicio.

Esto conduce a concluir que para garantizar la calidad de servicio en una arquitectura con colas virtuales a la salida se hace necesario disponer de un algoritmo capaz de asegurar la puntualidad del envío de paquetes almacenados entre los puertos.

2.5.2 Colas a la entrada y a la salida

[BDE+04] presenta un modelo de arquitecturas con colas de entrada y colas de salida. Dicho modelo balancea el ancho de banda para mantener las tasas de bits de flujos de datos constantes.

Lee y Kuo [LK05b] diseñan un algoritmo que emula las prestaciones de un conmutador con colas reales a la salida. En ambos casos, el gran problema es la necesidad de duplicar la velocidad de la matriz de conmutación, lo cual los hace poco aplicables por el elevado coste físico.

2.5.3 Colas de entrada y memoria interna

La aplicación de este modelo es especialmente interesante debido a las cualidades que se obtiene al aplicarlo con una matriz de conmutación interna tipo crossbar.

La aplicación de varias calidades de servicios implica la necesidad de colocar varias colas virtuales tanto en los puertos de entrada como en los elementos de conmutación de la matriz interna. Sin embargo el aumento de lógica no tanto como cabría esperar. En [CK04] se expone un método dinámico de ajuste que permite manejar múltiples clases con solo dos colas en cada elemento de conmutación. El coste de este método es una menor diferenciación entre las clases de servicios.

Otro enfoque distinto para garantizar diferentes calidades de servicio, es aumentar la inteligencia del algoritmo en las colas de entrada pudiendo reducir a tan solo una, el número de colas necesarias en los elementos de conmutación. Cabe destacar [YQW06], donde se usa un algoritmo llamado Priority Weighted Double Round Robin (PWDRR) en las colas virtuales de salida combinado con un segundo algoritmo denominado (Compensation Priority Round Robin CPRR) en las colas internas de la matriz crossbar. El resultado en un rendimiento próximo al de un conmutador con colas reales de salida y un algoritmo de planificación Weighted Fair Queuing (WFQ).

Por último, si se centra la atención en maximizar el throughput del sistema, hay estudios que presentan arquitecturas con colas de salida y memoria interna. [OMW06], se presenta el algoritmo GBSBF-LBF (Guaranteed Bandwidth Smallest Buffer First – Larger Buffer First). Este algoritmo se basa en dos claves: un mecanismo de créditos que asegura el reparto del ancho de banda y el control del nivel de ocupación de la matriz de conmutación interna para mejorar las prestaciones del conmutador. Como detalle final, el ancho de banda “sobrante” (no reservado) se reparte a posteriori usando un algoritmo que maximiza el throughput.

2.5.4 Colas a la salida

Una de las últimas aportaciones en la investigación para incluir calidad de servicio en un conmutador con colas de salida son las memorias PIFO (Push-In, First Out).

Cada paquete se inserta de forma ordenada (atendiendo a su prioridad y clase). La cola de salida transmitirá el primer paquete de la cola. El problema de este método es el coste de la ordenación de los paquetes en las memorias. En el mejor de los casos, insertar un elemento en la cola de salida tiene un coste de O(LogN), un valor elevadísimo para ser implementado en un conmutador de alta velocidad.

Algunas alternativas [WH07], proponen mejoras en el algoritmo de la PIFO usando inserción indexada (iPIFO).

Tal y como se muestra en [QLY+06] esta última propuesta es viable desde el punto de vista de la emulación, pero en ningún caso, abarcable en hardware.

3. El espacio. Características y herramientas.

A lo largo del presente trabajo se han presentado ciertas características que convierten al sector aero-espacial en uno de los sectores donde la ingeniería se pone más al límite. A la hora de desarrollar un sistema electrónico para un satélite son muchas las dificultades que hay que afrontar, dificultades que sólo están presentes en este entorno único, el espacio.

En este capítulo se mostrarán algunas de estas peculiaridades y las formas de afrontarlas.

El espacio es un entorno único. Cualquier sistema que no haya sido desarrollado específicamente para el espacio es un sistema inviable para funcionar allí. La atmósfera terrestre no solo protege a las personas de las inclemencias del espacio exterior sino que también absorbe multitud de partículas cósmicas que podrían dañar los sistemas electrónicos diseñados en tierra. Además, un factor de gran importancia es la temperatura. En la superficie terrestre la variación de la temperatura puede parecer grande para una persona (-30º - 50º) pero para un sistema electrónico es algo que se puede asumir con relativa facilidad. Sin embargo, en cuanto se abandona la protección de la atmosfera, el rango de temperaturas se amplía descomunalmente, pudiendo fundir cualquier circuito en cuestión de milisegundos o congelar cualquier relé, haciéndolo completamente inútil.

Aunque la vibración no es una característica provocada por el espacio sí es necesario tenerla en cuenta dado que los lanzadores actuales (Soyuz, Ariane, Atlas…) provocan altas vibraciones durante la etapa de lanzamiento y por ello todo sistema electrónico debe estar diseñado para soportarlas.

Otro aspecto de suma importancia son las radiaciones cósmicas. A medida que la tecnología ha ido evolucionando, el tamaño de las unidades lógicas de los sistemas electrónicos ha ido disminuyendo, favoreciendo así el desarrollo de sistemas más complejos en menor espacio y con menor peso. Sin embargo, esta tendencia hacia la miniaturización (actualmente se manejan tecnologías de nanómetros) hace que aparezcan efectos antes no observados. Éste es el caso de los SEU (Single Event upset). Debido a la importancia de este efecto, se dedicará una sección para describirlo.

Dentro de los sistemas electrónicos y sobre todo de los digitales cabe destacar una tecnología en particular, las FPGAs. Estos dispositivos, debido a su versatilidad,, están ganando poco a poco gran importancia en el sector, asumiendo poco a poco roles que en un principio eran impensables. Cada año, los sistemas implementados en las FPGAs aumentan su complejidad y absorben mayor protagonismo dentro de las tarjetas electrónicas que conforman una caja de vuelo. Debido a su creciente importancia se le dedicará un sección en particular.

3.1 FPGAs

FPGA (Field Programable Gate Array), es un circuito integrado que está pensado para ser configurado por el cliente. Utilizando un lenguaje de descripción de hardware, como el HDL (Hardware Description Language) se puede desarrollar dentro de uno de estos chips sistemas de altísima complejidad.

Una de sus principales características es la posibilidad de reprogramar su funcionalidad (esto no es aplicable a todos los modelos de FPGA) permitiendo así flujos de diseño recurrente, que simplifican el proceso y abarata los costes de desarrollo.

Al tratarse de un sistema puramente hardware la paralelización de procesos permite llevar a cabo gran cantidad de operaciones y cálculos de forma simultánea (al contrario que un microprocesador que ejecuta las instrucciones de forma secuencial).

Algunos de estos dispositivos llevan asociados una memoria donde se almacena la descripción de la funcionalidad implementada, de tal manera que en cualquier momento puede ser corregida, actualizada o modificada sin necesidad de soldar o desoldar un solo pin.

Sin embargo, la presencia de memorias, de elementos de almacenamiento (registros, Look Up Tables, CLBs…) las convierten en blancos fáciles para los errores provocados por las radiaciones cósmicas. Estos efectos y las diversas técnicas para mitigarlos serán descritos en el siguiente apartado.

Todas estas características hacen de las FPGAs elementos ampliamente usados en los satélites y en el caso del presente estudio, las convierte en un perfecto candidato para la implementación, a bajo nivel, del conmutador para el espacio.

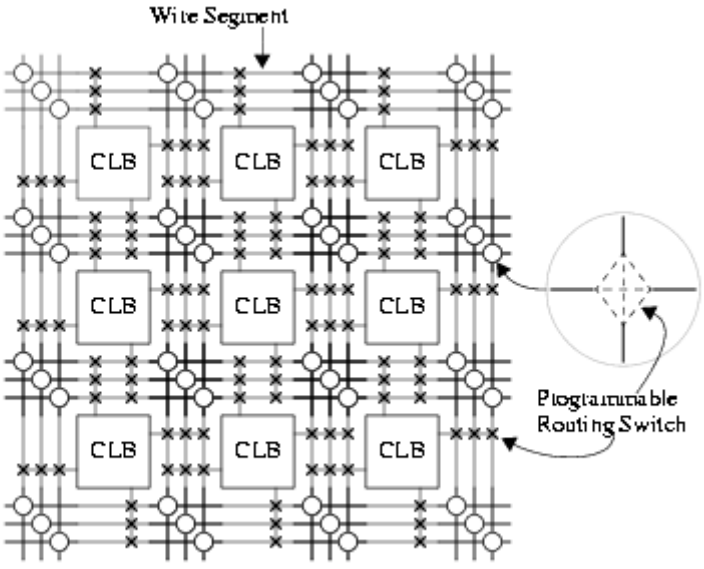

Figura 3.1: Arquitectura genérica de una FPGA

3.2 Radiaciones cósmicas y sus efectos. SEUs

La atmósfera terrestre filtra gran parte de las radiaciones que circulan por el espacio. Son muchas y diversas las fuentes de estas radiaciones y peligrosos y muy cuantiosos los efectos negativos que pueden tener sobre un sistema digital.

Cuando una partícula cargada incide sobre un sistema digital puede atravesarlo sin causar ningún efecto o causar destrozos incalculables. Si el sistema tiene una densidad electrónica baja, la probabilidad de incidencia de una partícula cargada sobre un elemento susceptible a la radiación es también baja. Además cuanto mayor es el umbral del sistema menor será el efecto del rayo cósmico.

Sin embargo, como se ha descrito con anterioridad, los sistemas electrónicos tienden a hacerse cada más pequeños y se busca que su consumo sea el menor posible. Este entorno es el perfecto para la aparición de SEUs (Single Event Upset).

Cuando una partícula pesada incide sobre un elemento de memoria (registro) puede provocar que el valor almacenado en ese registro cambie. Este inesperado cambio de signo es lo que se conoce como SEU. Si el SEU se produce en un elemento transitorio, el efecto general será también transitorio o incluso puede pasar desapercibido. Pero si se produce un SEU en una lógica de decisión importante, podría llegar a apagar una batería, ordenar el arranque de un motor o desactivar el sistema de comunicaciones.

Para evitar estos efectos, existen diferentes técnicas de mitigación, corrección o eliminación. La triple redundancia modular (Triple Modular Redundancy – TMR), los sistemas de votación, EDAC (Error Detection And Correction), la técnica del scrubbing o el encapsulado Rad-Hard son algunos ejemplos.

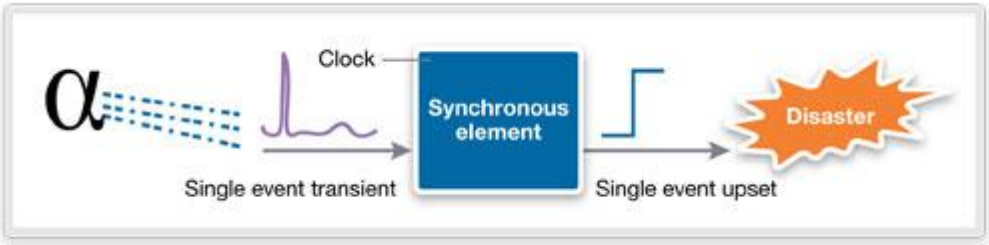

Figura 3.2: Aparición de un SEU sobre un registro síncrono

4. Arquitecturas comerciales

Obviamente, este trabajo no es el primero que plantea la necesidad de un sistema de conmutación para los satélites. Son muchos los que han planteado esta necesidad previamente y existen distintos dispositivos comerciales que implementan de una u otra forma la funcionalidad que se expone en el presente estudio. Como ya se ha comentado antes, el sector aero-espacial es un sector industrial muy restringido. Por ello la información sobre los dispositivos existentes es limitada y en algunos casos puede que hasta sesgada. Sin embargo, las arquitecturas que se van a exponer sirven como punto de partida para estipular las necesidades del modelo aquí propuesto y para poder situarlo dentro del mercado existente.

4.1 ATMEL AT7910E

Este dispositivo de la empresa de microchips ATMEL. Presenta un router basado en el protocolo SpaceWire. Implementa 8 puertos de entrada/salida más dos puertos dedicados que ofrecen un servicio diferenciado tipo FIFO.

Entre sus características más destacadas se encuentran una tasa de transferencia de 200 Mbps por puerto, manejo de prioridades y sin restricción ITAR (International Traffic in Arms Regulations) [PJB08] para la exportación. En la figura 4.1 [ATM13] se muestra un esquema de su arquitectura interna.

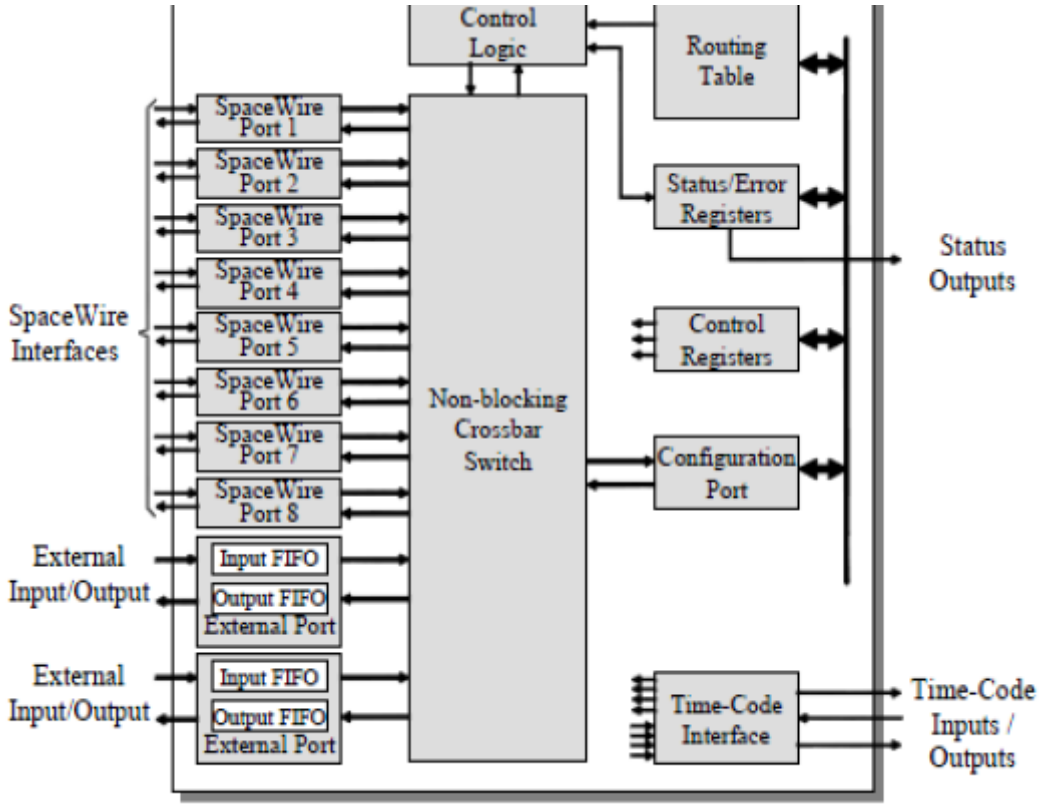

Figura 4.1: Diagrama de bloques del router SpaceWire de ATMEL

4.2 RT-SPW ROUTER

El fabricante americano de FPGAs Microsemi (antes ACTEL) presenta una solución similar al router de ATMEL. 8 puertos de entrada/salida con dos más dedicados a estructuras de tipo FIFO. Una tasa de transferencia de 200 Mbps y manejo de prioridades. Uno de sus principales problemas es la restricción impuesta ITAR, impuesta por el gobierno de los Estados Unidos al ser tecnología "sensible". En la figura 4.2 [AER12] se muestra un esquema de su arquitectura interna.

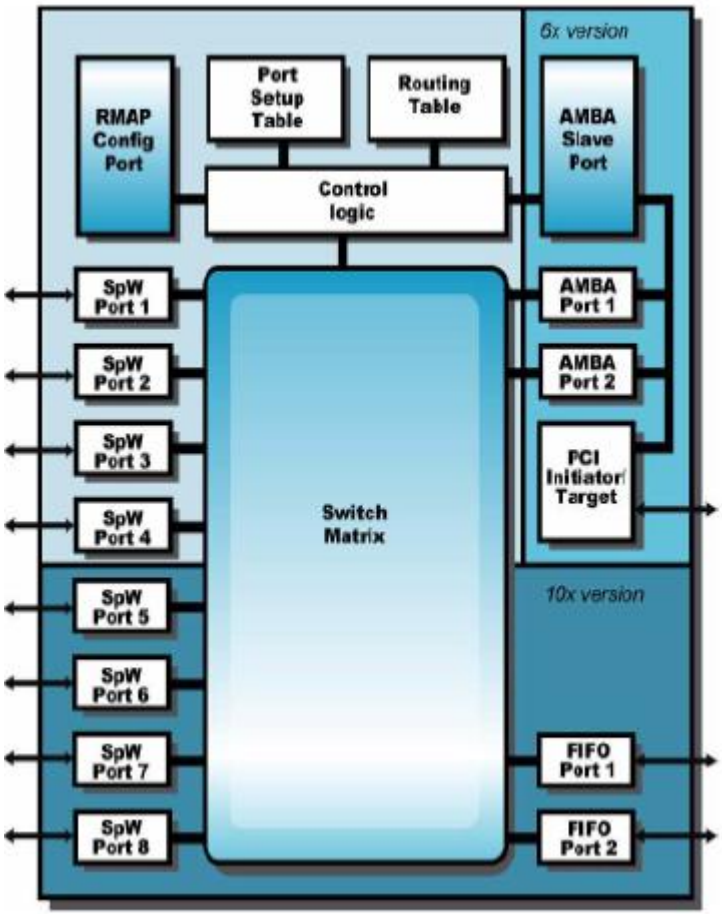

Figura 4.2: Diagrama de bloques del router SpaceWire de Microsemi

4.3 UT200SpW4RTR

La empresa americana AEROFLEX presenta una arquitectura de características algo distintas a lo planteado hasta ahora. Aunque la tasa de transferencia es igual a las anteriores (200 Mbps por puerto) este dispositivo solo dispone de 4 puertos de entrada/salida más uno dedicado a la política FIFO. Al igual que los anteriores, soporta trato de prioridades Como en el caso del dispositivo de Microsemi, éste material tiene restricciones de exportación (ITAR). En la figura 4.3 [AERO13] se muestra un esquema de su arquitectura interna.

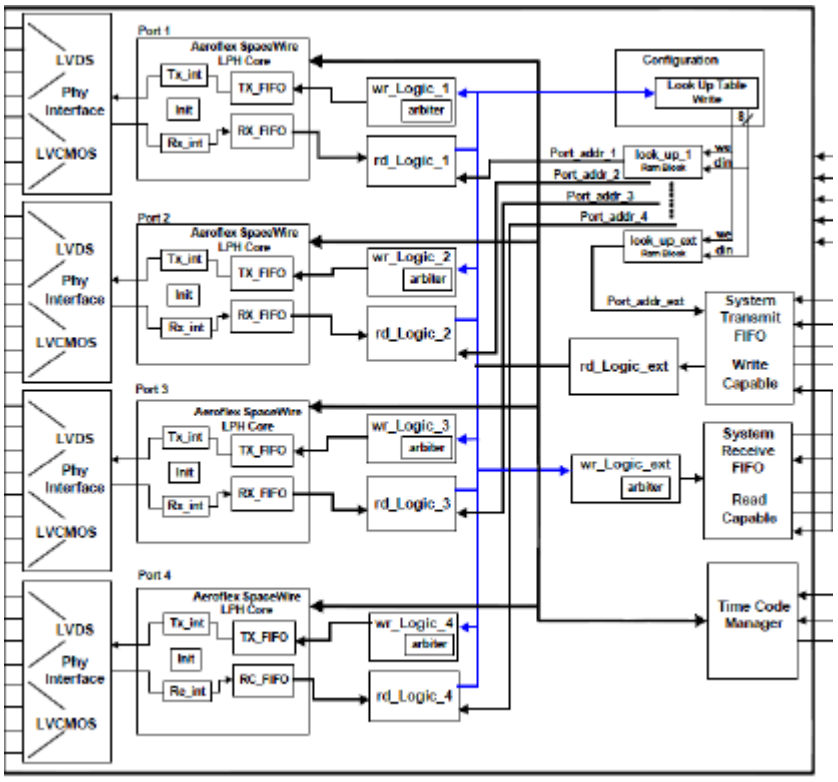

Figura 4.3: Diagrama de bloques del router SpaceWire de AEROFLEX

4.4 BAE SpW router

La empresa británica de sistemas para la defensa y el espacio dispone de un sistema de rutado propio. En este caso, el dispositivo ofrece 16 puertos de entrada/salida SpaceWire de 256 Mbps cada uno. Además dispone de un puerto dedicado de tipo SPI (Serial Peripheral Interface) y otro para el acceso directo a una memoria RAM externa. Se desconocen las restricciones de exportación de este dispositivo. En la figura 4.4 [BAE11] se muestra un esquema de su arquitectura interna.

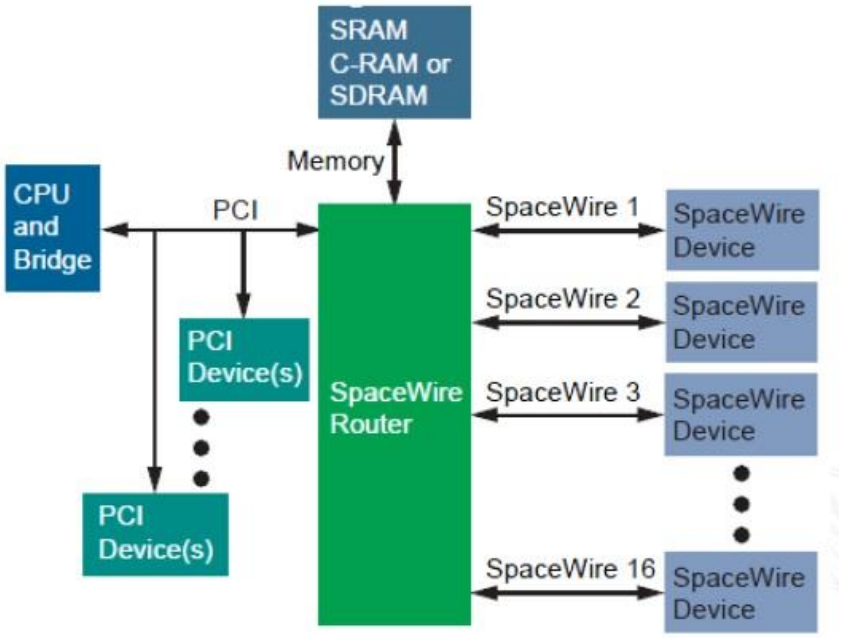

Figura 4.4: Diagrama de bloques del router SpaceWire de BAE

5. Planificación de tráfico

5.1 Introducción

Por regla general, en una red, los flujos de datos se multiplexan y compiten entre ellos por los recursos de la propia red. Por ello surgen políticas de gestión de la red que clasifican los diversos flujos de datos, priorizando el acceso a los recursos compartidos. Concretando en el caso del conmutador, el algoritmo de planificación es quien se encarga de decidir qué paquete sale por un determinado puerto y en qué momento. Esta acción determina en gran medida parámetros como el retardo de cada uno de los paquetes (tiempo transcurrido desde que entra en el conmutador hasta que sale por el puerto de salida correspondiente) y el ancho de banda. Además, cada flujo de datos tiene necesidades distintas (diferentes calidades de servicio), esto es, para cada paquete hay que tener en cuenta la probabilidad de pérdida, la latencia o el ancho de banda de forma distinta al resto.

La cualificación del tráfico de datos que circula por el conmutador se puede reducir a un principio básico de funcionamiento: la asignación de recursos particularizada repercute inevitablemente en el comportamiento general del sistema, esto es, si un flujo determinado exige un cierto ancho de banda, es obvio que otro flujo de datos verá reducido su asignación. Esta ley de conservación (anunciada por Kleinrock en 1975 [Kle75]) se utiliza para estudiar las características del conmutador propuesto en este trabajo.

5.2 Modelos de tráfico

Antes de entrar en detalles con el estudio de las características de la arquitectura de conmutación y del algoritmo propuesto, es necesario hacer un pequeño inciso en la distribución de tráfico utilizada.

Para caracterizar el sistema propuesto se ha elegido la distribución de tráfico más referenciada en la bibliografía consultada, el modelo de Bernoulli [AOS+93][McK95].

5.2.1 Tráfico uniforme

La generación de tráfico uniforme se basa en un proceso de llegadas de Bernoulli. Para caracterizar modelos de conmutación, ante tráfico uniforme, en los ciclos de decisión del reloj se genera un número pseudo-aleatorio para determinar si se genera o no un paquete de datos. En caso de no generar paquete alguno, esta generación es sustituida por un periodo de inactividad. El paquete generado tendrá distintos destinos e incluso una dirección multicast.

En función de estas acciones los destinos son elegidos de manera uniforme. En caso de disponer de diferentes clases de tráfico, la clase asignada es seleccionada de forma aleatoria. El siguiente paso es el encargado de seleccionar el tamaño del paquete. Al finalizar la transmisión del paquete generado se espera un nuevo ciclo de reloj y se repite el proceso descrito.

5.3 Validación del modelo de alto nivel

En esta sección se presentan los resultados obtenidos en la simulación de distintos conmutadores descritos en la bibliografía. Antes de caracterizar el modelo final se han realizado una serie de simulaciones con el fin de validar el sistema en sí. Para ello se ha desarrollado un modelo de alto nivel que permite simular las características de los modelos y estimar sus prestaciones, facilitando así una descripción funcional detallada. Estos modelos de alto nivel se han desarrollado usando el lenguaje programación C con la ayuda de las librerías CSIM [CSIM]. Este software comercial, permite emular el comportamiento de un sistema hardware (paralelismo). Además, facilita una serie de herramientas matemáticas y estadísticas que ayudan al análisis del modelo planteado.

Para estas simulaciones se han elegido las siguientes arquitecturas:

- OQ: Conmutador con colas a la salida de tamaño infinito, con política de planificación de tráfico FIFO y paquetes de tamaño variable. Esta arquitectura va a representar el límite inferior e ideal en términos de retardo y que supone el objetivo al que aspira todo dispositivo.

- VOQ: Conmutador con colas virtuales de salida, con política de planificación iSLIP (4 iteraciones) y paquetes de longitud variable.

- MOQ: Conmutador múltiples colas de salida, algoritmo de planificación Round Robin y paquetes de longitud variable. Éste es una simplificación de la arquitectura propuesta (sin QoS).

Para todas las arquitecturas se consideran 8 puertos de entrada y 8 puertos de salida. En las figuras 5.1, 5.2 y 5.3 se muestran los resultados de las simulaciones

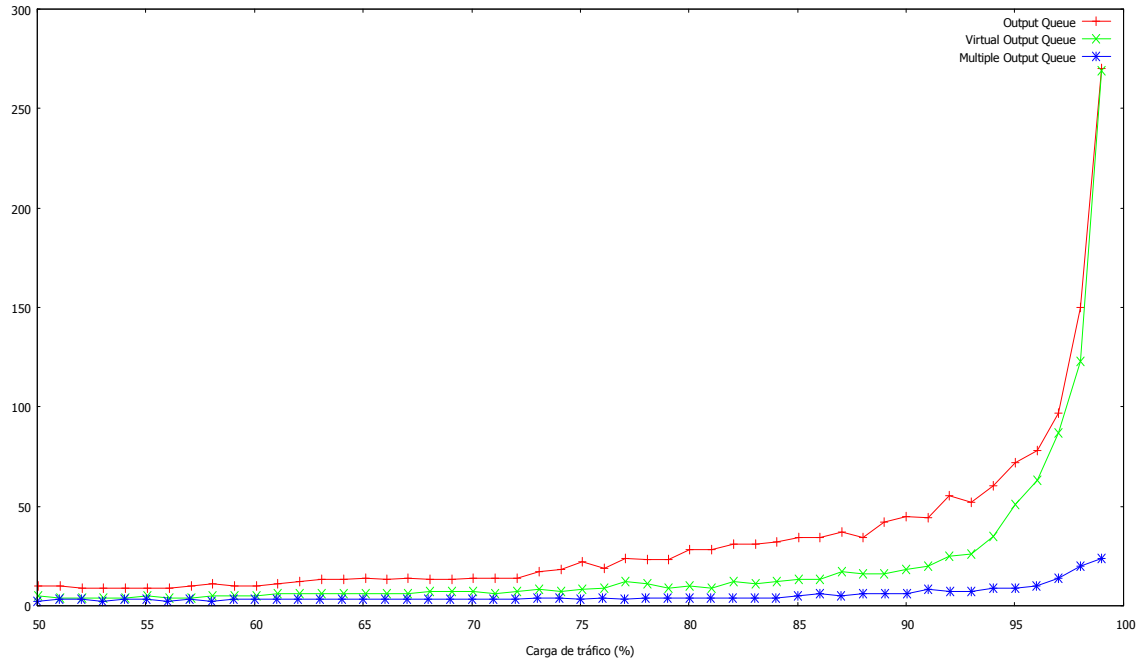

Figura 5.1: Tamaño máximo de las colas de salida con tráfico uniforme en los conmutadores OQ, VOQ y MOQ

La figura 5.1 muestra el tamaño de las colas de salida de las tres arquitecturas propuestas y su evolución en función de la carga de tráfico del conmutador. Como era de esperar el modelo MOQ presenta un tamaño menor de colas debido a la existencia de varias colas por cada puerto de salida.

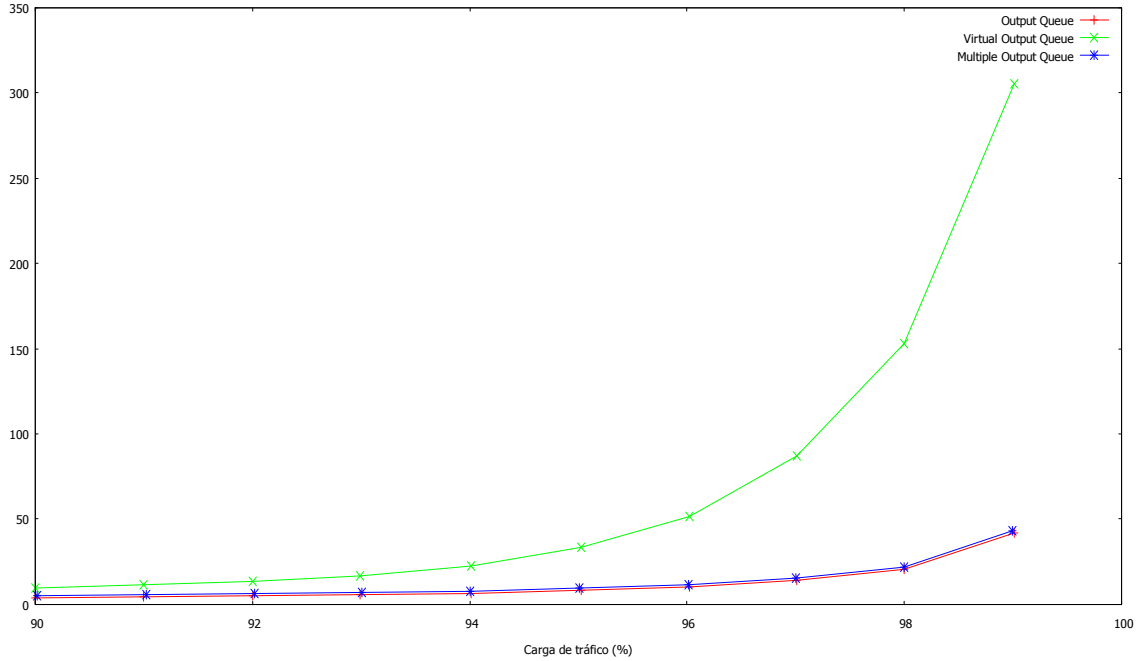

Figura 5.2: retardo medio de encolado con tráfico uniforme en los conmutadores OQ, VOQ y MOQ

La figura 5.2 muestra el retardo de encolado de las tres arquitecturas propuestas y su evolución en función de la carga de tráfico del conmutador. Como era de esperar el modelo OQ tiene el retardo más bajo y el MOQ acerca su comportamiento al ideal. El modelo VOQ, tal y como se comentó en apartados anteriores, utiliza un algoritmo de planificación iSLIP. Este basa su funcionamiento en la filosofía Round Robin con handshake de forma iterativa. Por este motivo, pueden ocurrir que el número de iteraciones sea insuficiente para enlazar todos los puertos de entrada o salida, o que la elección entre ellos no sea óptima.

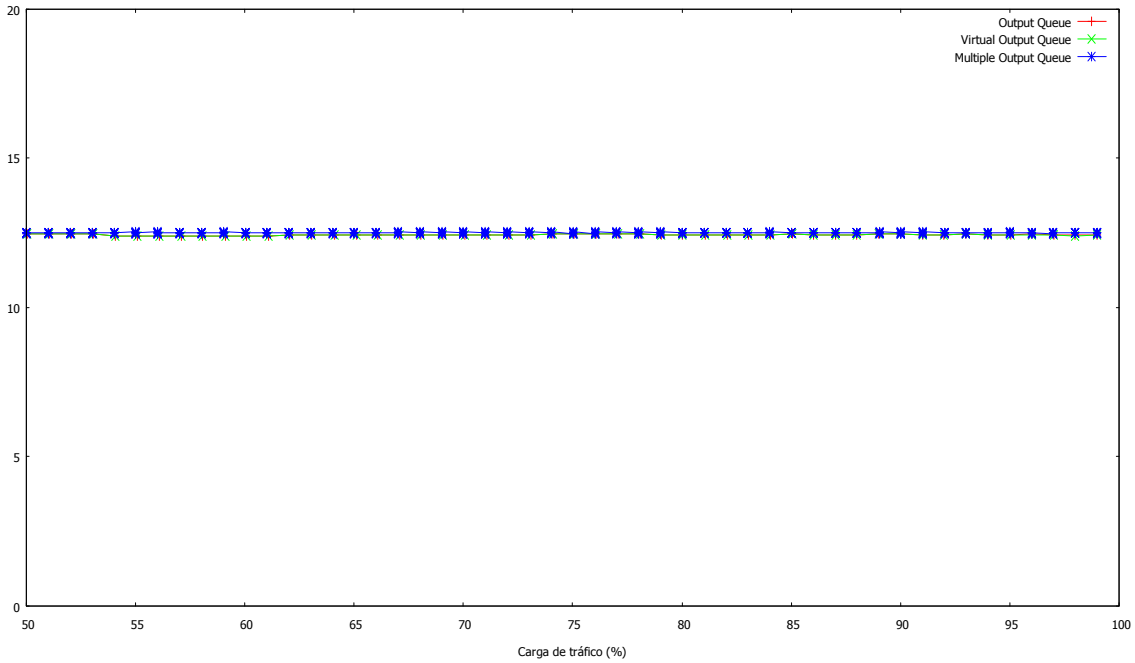

Figura 5.3: Ancho de banda medio asignado al puerto de salida '0' con tráfico uniforme en los conmutadores OQ, VOQ y MOQ

En esta figura se muestra el ancho de banda asignado a cada puerto de salida. Como era de esperar las tres arquitecturas se comportan de igual manera (debido a la no existencia de QoS). El ancho de banda observado en cada puerto de salida (en la gráfica esta particularizado para el puerto '0') es un 12'5% (1/8) dado que en los tres casos tenemos 8 puertos de salida.

5.4 Descripción del modelo propuesto

El objetivo final del presente trabajo es encontrar un sistema de rutado capaz de satisfacer las necesidades de las comunicaciones internas de un satélite. De entre todos los diferentes algoritmos estudiados no se ha encontrado ninguno que dé cabida a todas las características expuestas:

- RR: es sencillo de implementar pero no admite paquetes de longitud variable.
- DRR: admite paquetes de longitud variable pero no está preparado para garantizar QoS.
- WRR: garantiza diferentes anchos de banda pero no admite paquetes de longitud variable ni tampoco la posibilidad de ofrecer diferentes calidades.
- PWRR: no admite paquetes de longitud variable.
- GPS: necesita demasiada complejidad computacional.

Además, ninguno de ellos tiene la posibilidad de implementar tráfico por clases. Sin embargo el estudio de estos algoritmos resulta de gran utilidad. Se observa que todos aquellos algoritmos basados en la filosofía Round Robin mantienen una gran simplicidad de implementación, lo que facilita el paso del algoritmo a una plataforma hardware.

Tomando esta idea como punto de partida, en este apartado se propone un algoritmo de planificación basado en Round Robin, que garantice todas las QoS definidas.

En el apartado siguiente se describe en detalle el algoritmo de planificación utilizado y se presenta un estudio de las características de éste.

5.4.1 Descripción del algoritmo de planificación

Las características de un algoritmo de planificación son tan numerosas como heterogéneas. Por ello, a la hora de elegir una política de planificación hay que tener en cuenta las necesidades de la red a fin de conseguir un rendimiento satisfactorio. Zhang en [Zha95] enumera las características más representativas:

- **Eficiencia**: a fin de mantener los requisitos del sistema de conmutación es necesario disponer de un sistema de control de admisión de conexiones (Call Admision Control – CAC) que permita limitar la carga de tráfico garantizado. A mayor eficiencia del algoritmo mayor será el número de conexiones que se pueden admitir y mayor el uso de los recursos de red.
- **Protección**: un algoritmo debe proteger las conexiones con buen comportamiento de tres fuentes de variabilidad: conexiones con mal comportamiento, fluctuaciones de carga de red y tráfico sin calidad de servicio.
- **Flexibilidad**: debido a la amplia variedad servicios el algoritmo de planificación ha de ser flexible a la hora de garantizar diferentes retardos, asignar distintos anchos de banda y asegurar probabilidades de pérdidas individualizadas.
- **Simplicidad**: la simplicidad de un algoritmo repercute principalmente en su escalabilidad. Este es un aspecto a tener en cuenta si el algoritmo en cuestión pretende ser implementado en hardware, como es el caso.

5.4.2 Características específicas del soporte de QoS

Siguiendo la filosofía de Servicios Diferenciados [BBC+98] es necesario garantizar la existencia de varias categorías de flujos de datos llamadas clases, Sin embargo, esto no es suficiente para alcanzar el objetivo de QoS necesario en un router para un satélite. A fin de aumentar la eficiencia y la flexibilidad de la planificación se proponen las siguientes clases de QoS:

- **Clases de tráfico:** cada paquete es clasificado atendiendo a sus necesidades, de tal forma que todos los paquetes de la misma clase reciben el mismo servicio.
- **Peso**: este parámetro es una representación de la proporción de ancho de banda total asignada a una clase de tráfico. Si los paquetes fueran de longitud fija, la relación entre el ancho de banda y el número de paquetes sería directa. En el caso del modelo propuesto los paquetes son de longitud variable lo que incremente la complejidad del algoritmo. El algoritmo DRR [SV96] es un ejemplo que se adapta a esta necesidad.
- **Prioridades:** para reducir o incrementar el retardo medio de una clase en particular es necesario poder asignar prioridades. Este hecho no debe impedir que el planificador mantenga la asignación del ancho de banda en función de los pesos, como se puede observar en el algoritmo PQWRR [MMW01].

- **Paquetes de longitud variable:** aunque existen algoritmos como WRR o DRR que manejan diferentes clases estos no son capaces de lidiar con la combinación de diferentes prioridades, clases, pesos y paquetes de longitud variable.

5.4.3 Características propias del algoritmo propuesto

Sobre las características propuestas se planten tres consideraciones adicionales:

- **Servicio conservador**: Siempre que existan paquetes almacenados su transmisión se prioriza con el fin de maximizar el throughput del sistema.
- **Paquetes de longitud variable**: las propuestas basadas en sellos de tiempo, como el algoritmo WFQ [PG93][DKS89], llevan asociado un gran coste hardware. La alternativa que se presenta en este trabajo usa un mecanismo basado en créditos obtenidos mediante la relación entre la longitud de los paquetes y el ancho de banda asignado.
- **Marco de tiempo:** un marco de tiempo es un período de tiempo en el que el planificador mantiene los pesos y las prioridades de forma justa. Así, cada clase de tráfico reserva una porción del marco de tiempo en función de su peso mientras que el orden de salida depende principalmente de la prioridad de la clase. En caso de algún desajuste en el marcos (todas las clases de tráfico consumen su porción o existen clases vacías) el marco se resetea.

5.4.4 Descripción funcional

Las características hasta ahora descritas componen un algoritmo de una elevada complejidad. Para abordar la problemática se decide descomponer el funcionamiento del algoritmo en dos mecanismos uno de control de retardo y otro de control de ancho de banda.

El planificador selecciona, de entre las clases disponibles, la de máxima prioridad. Ante igualdad de prioridades se usa un Round Robin para “desempatar”. El segundo paso selecciona el flujo de datos teniendo en cuenta la información enviada con anterioridad. En la figura 5.4 se muestra la filosofía jerárquica descrita, seleccionando primero la clase y después la fuente de tráfico.

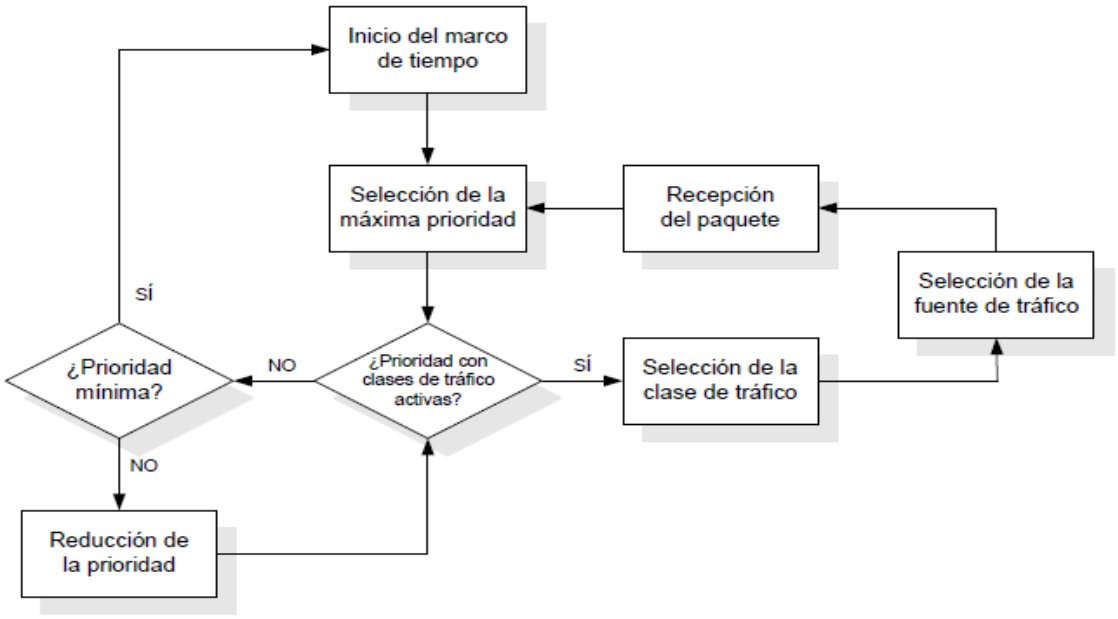

Figura 5.4: Flujo de control del algoritmo de planificación

El mecanismo de control se basa en la asignación de créditos. Cada clase tiene asignado un crédito, cada vez que trasmite un paquete consume parte de ese crédito proporcionalmente con el tamaño del paquete transmitido. Cuando la clase agota el crédito deja de poder transmitir. Una vez que todas las clases se queden sin crédito el marco de tiempo se resetea y vuelve a asignar crédito a cada clase en función del peso de cada una de ellas.

a. Selección de la clase de tráfico

Como se ha expuesto en el apartado anterior el algoritmo propuesto se divide en dos fases. En la primera fase selecciona la máxima prioridad de entre las existentes, y se elige una clase de tráfico activa usando la política Round Robin. Se entiendo por una clase activa, aquella que tiene la prioridad seleccionada, no está vacía y tiene crédito. El concepto de crédito es directamente traducible por ancho de banda. Así si una clase tiene crédito significa que no ha consumido el ancho de banda asignado en el marco de tiempo actual. Si la prioridad elegida no tiene ninguna clase con crédito entonces e reduce la prioridad. La figura 5.5 muestra un esquema del funcionamiento del algoritmo:

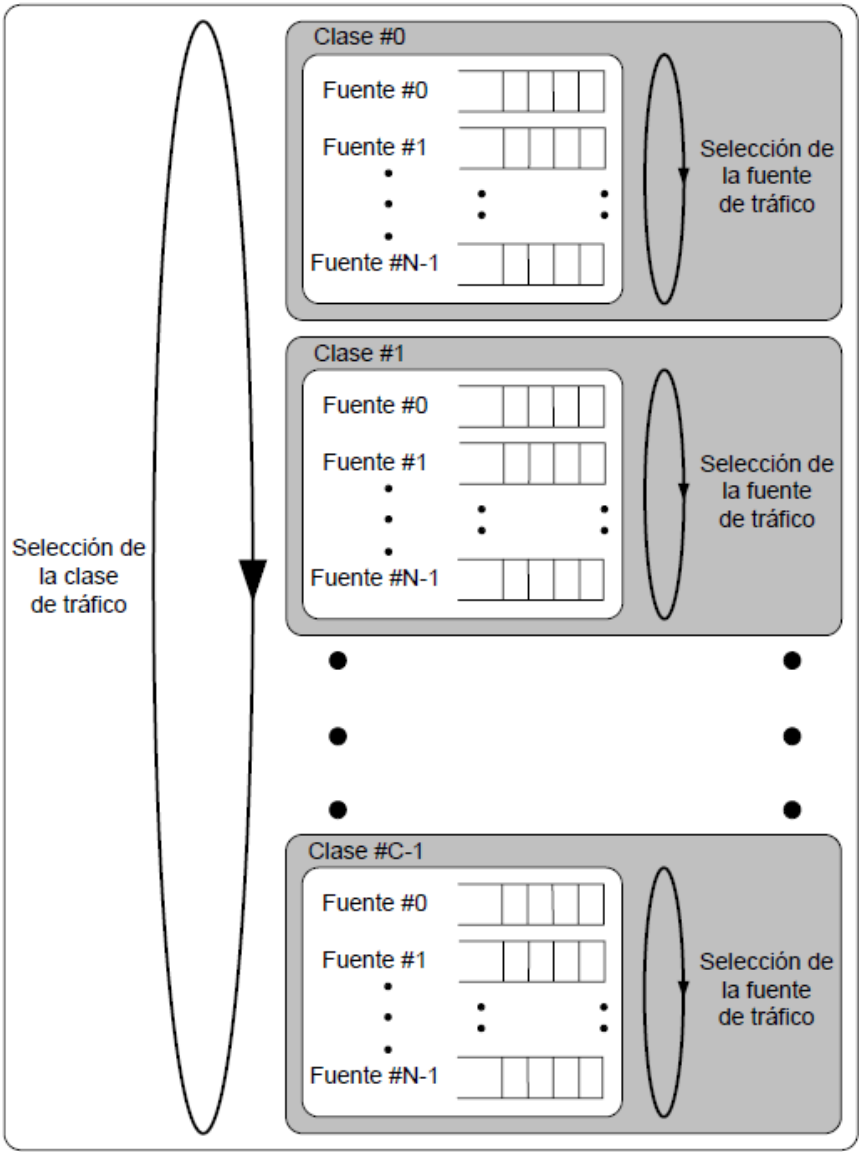

Figura 5.5: Estructura de almacenamiento del algoritmo de planificación.

b. Selección de la fuente de tráfico

Una vez seleccionada la clase de tráfico a transmitir hay que seleccionar la fuente de tráfico. Para ello se comprueba cual de las acumuladas no está vacía y de entre ellas se elige la fuente que menos paquetes haya enviado hasta el momento.

c. Control de ancho de banda

Hasta que el paquete no llega al planificador, éste desconoce el tamaño. Por ello, una vez extrae el paquete de la memoria, se encarga de actualizar los registros de los créditos de las clases tráfico así como la cantidad de tráfico enviada por cada clase y cada flujo.

d. Marco de tiempo

Nada más inicializar el marco de tiempo, se calcula el valor de los créditos en función de los pesos asignados a cada clase. En realidad, estos pesos equivalen al ancho de banda asignado para un marco de tiempo determinado. La filosofía del marco de tiempo permite

que, independientemente de la configuración de pesos, el envío de un paquete de longitud máxima y menor peso (peor de los casos) esté garantizado.

La siguiente ecuación muestra el cálculo valor del crédito asignado a una clase 'n':

$$Valor\ del\ crédito\ _{Clase\ de\ tráfico\ n} = L_{MAX}\ \frac{Peso\ _{Clase\ de\ tráfico\ n}}{Peso\ _{MIN}} + C_n,$$

Donde Peso clase de tráfico n es el peso asignado a la clase n, Peso MIN, es el peso mínimo resultante de la comparación de todas las clases, L_MAX es la longitud máxima de todos los paquetes y C n un factor de corrección proveniente del marco anterior.

El cálculo del crédito se realiza cuando no se puede elegir ninguna cola de tráfico debido a que todas se han vaciado o ninguna tiene crédito. Si se considera que las colas del conmutador siempre almacenan algún paquete la condición de inicio del marco de tiempo es el valor del crédito de las clases de tráfico. Consecuentemente, la longitud máxima del marco de tiempo es:

$$Longitud\ _{MT} = \sum_{n=0}^{n=C-1} (\ L_{MAX}\ \frac{Peso\ _{Clase\ de\ tráfico\ n}}{Peso\ _{MIN}} + C_n\ ),$$

Siendo C el número total de clases soportadas por el modelo. Según esta ecuación, la longitud del marco de tiempo es resultado de la relación entre los pesos de las clases sobre la base de la longitud máxima permitida del tamaño del paquete. Por ello se puede asegurar que la longitud del marco de tiempo se adapta a la configuración para garantizar el reparto de anchos de banda.

Un factor determinante para el algoritmo de planificación es el significado del factor de corrección Cn. El planificador solo sabe el nivel de ocupación de las colas y el crédito disponible de cada clase. La problemática que genera el hecho de elegir paquetes de longitud variable es que el crédito disponible no suele coincidir con la carga útil. Al final del marco de tiempo, suele aparecer un valor Cn negativo que representa la cantidad de bytes que una clase ha transmitido en el marco anterior sobrepasando el crédito asignado. Éste factor de corrección se le aplica a la clase en el siguiente marco de tiempo compensando el exceso del anterior, asegurando así, el reparto del ancho de banda. En el caso de que la clase esté vacía, el factor de corrección se desecha.

5.5 Caracterización del modelo propuesto

En este capítulo se persigue conocer el rendimiento del conmutador con diferentes configuraciones a fin de demostrar el soporte a las QoS anteriormente descritas. Esto es

especialmente importante desde el punto de vista del satélite pues es necesario poder diferenciar los distintos tipos de comunicaciones adaptando cada línea a las necesidades de las unidades que conectan.

En las simulaciones que se van a presentar se modifica la asignación del ancho de banda y de las prioridades pero se mantiene fijo el número de clases de tráfico en 8. Observando el capítulo 3 del presente trabajo, se pueden apreciar la variabilidad en el número de puertos que considera óptimo cada fabricante. En el presente estudio se ha considerado que un número medio de puertos y de clases puede aportar información más representativa.

Una de las principales características del protocolo SpaceWire es la posibilidad de tener paquetes de tamaño variable.

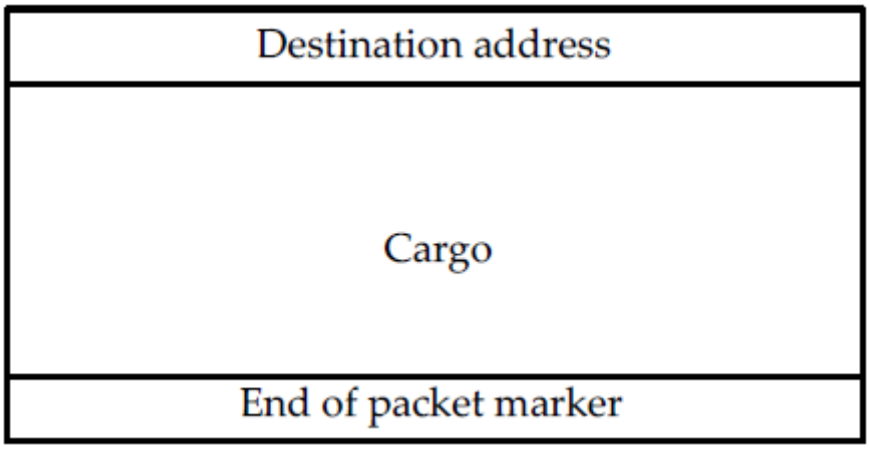

Figura 5.6: formato del paquete SpaceWire [EC12]

El conmutador descrito anteriormente, así como el modelo de simulación que implementa el algoritmo de planificación, contempla esta necesidad. Por este motivo todas las simulaciones se han realizado variando la carga útil siguiendo una distribución uniforme entre 1 y 248 (se estiman paquetes con un tamaño máximo de 256 bytes, 248 de carga útil y 8 bytes de cabecera).

Por último, a fin de garantizar la fiabilidad de los datos obtenidos, se fuerza la obtención de 10000 paquetes por cada fuente/clase de tráfico. Este valor garantiza un intervalo de confianza del 95% con una precisión del 2%.

5.5.1 Caso 1: Punto de partida

Este es el punto de partida de la caracterización. Como tal se han planteado las condiciones más favorables posibles: sin QoS. Todas las clases son tratadas de igual manera, sin diferenciar ni prioridades ni anchos de banda. El resultado es un ancho de banda y un retardo similar para todas las clases.

Ver figura 5.7 y figura 5.8

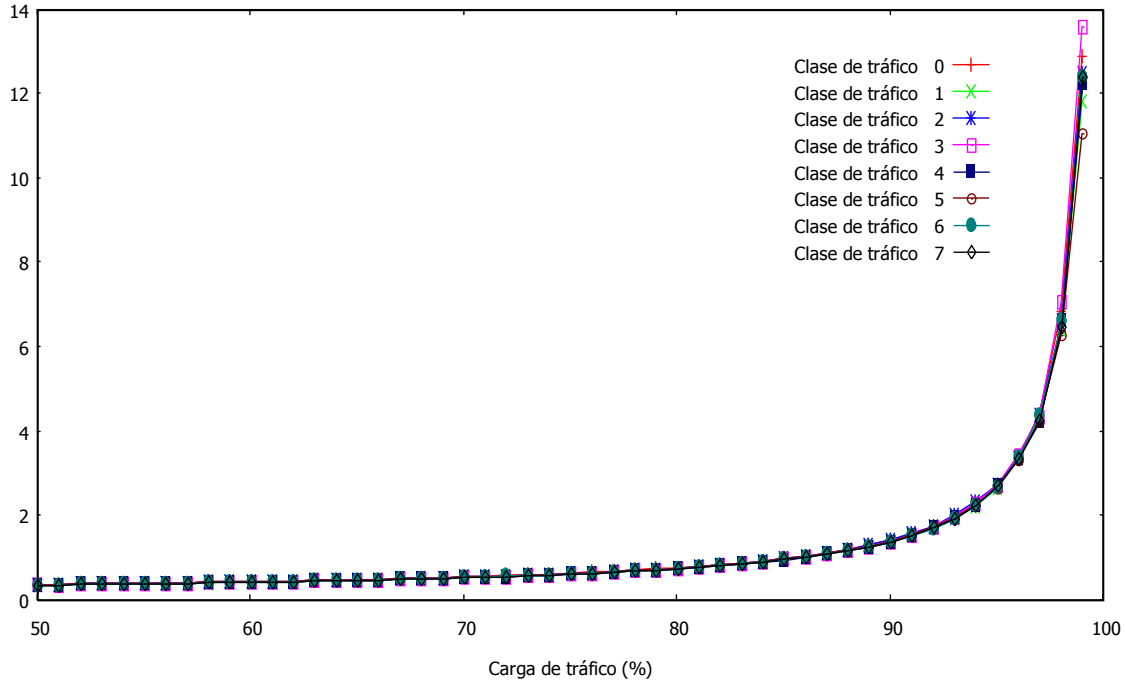

Figura 5.7: Retardo medio de encolado para el caso 1

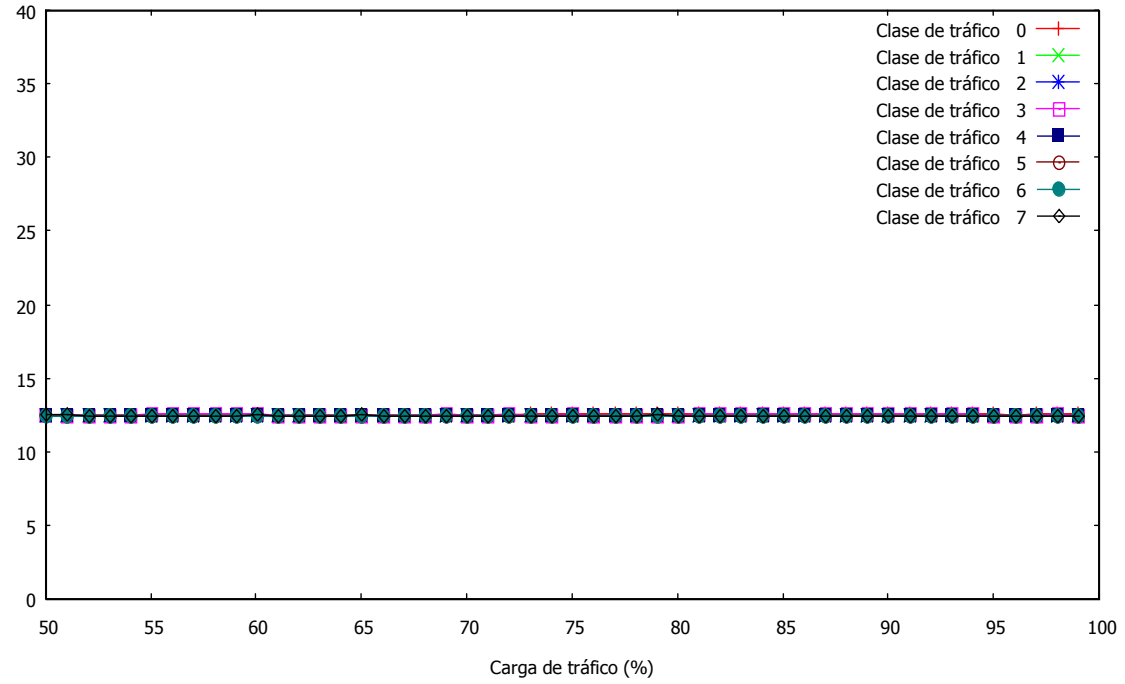

Figura 5.8: Ancho de banda asignado a cada clase para el caso 1

5.5.2 Caso 2: Control del ancho de banda

El objetivo de este supuesto es comprobar el efecto en el retardo, que tiene el ancho de banda asignado a cada clase. Para ello se configura el modelo con diferentes anchos de banda, según las clases pero sin ninguna prioridad. Lo que se espera es que si el ancho de banda es relativamente grande con respecto al resto de las clases su retardo sea menor que el de las demás. Si el ancho de nada es reducido el retardo, consecuentemente, aumentará.

Los valores de ancho de banda asignado son: 2%, 8%, 16% y 24%. Al haber 8 clases diferentes estos anchos de banda se asignan dos a dos. Como se preveía, la figura 5.9 muestra un retardo más reducido para las clases con mayor ancho de banda.

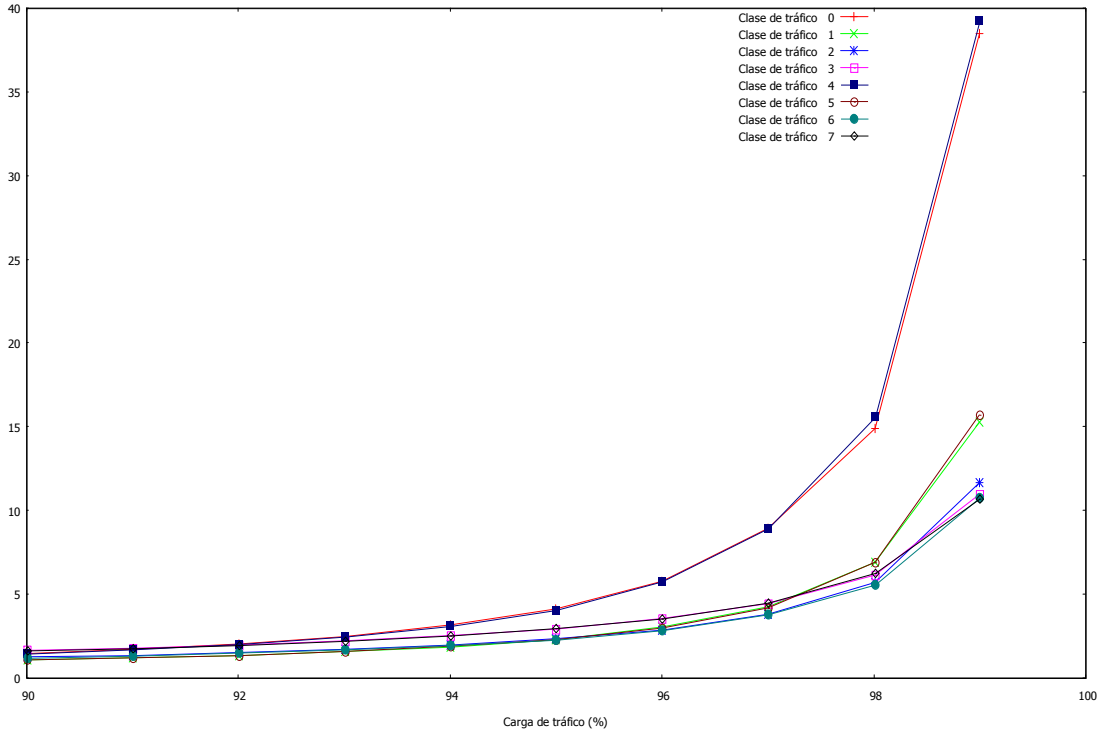

Figura 5.9: Retardo medio de encolado para el caso 2

Como se preveía, la agrupación del as clases responde a las cuatro configuraciones de ancho de banda enunciadas. Dos de ellas son claramente diferenciables, aquellas con menor ancho de banda y por ello mayor retardo. Las otras dos configuraciones agrupan el resto de las clases con poca distancia entre ellas, aunque si es apreciable pequeñas diferencias entre ellas.

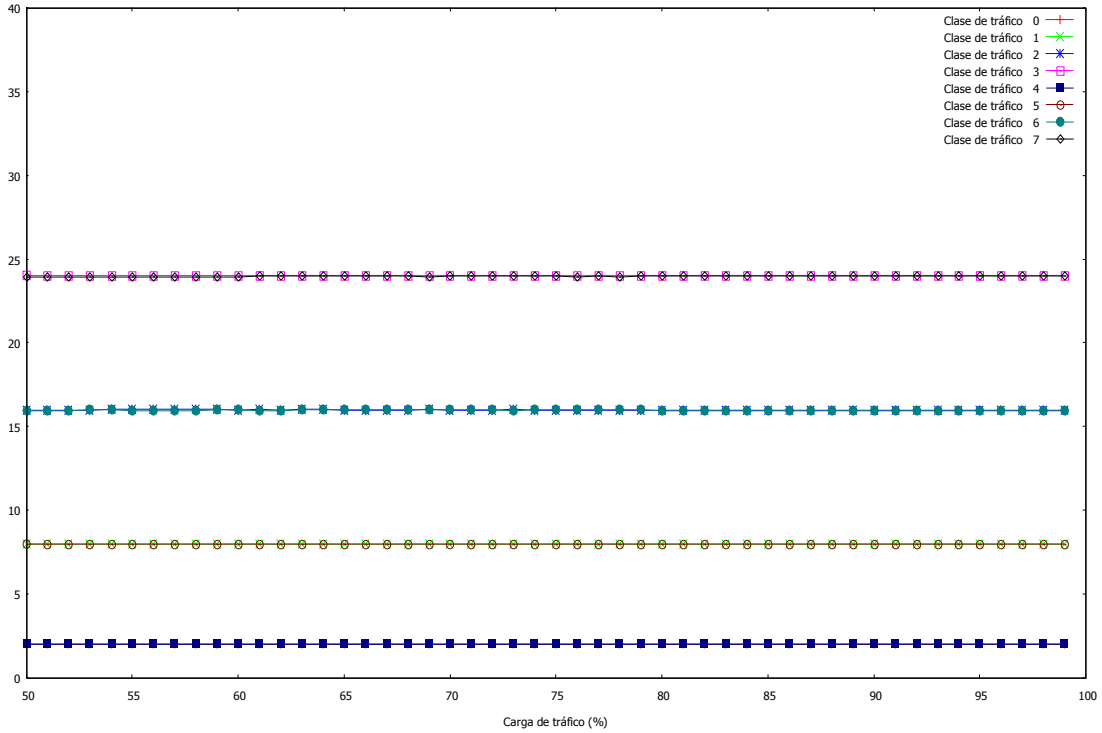

Figura 5.10: Ancho de banda asignado a cada clase para el caso 2

5.5.3 Caso 3: Control del retardo

El objetivo de este supuesto es comprobar el efecto de las prioridades en el retardo de transmisión de los paquetes de una clase, asumiendo dos prioridades distintas. Observando la figura 5.11 se puede ver claramente la diferencia entre las clases asignadas a una u otra prioridad.

La figura 5.12 muestra que el ancho de banda no sufre ninguna variación.

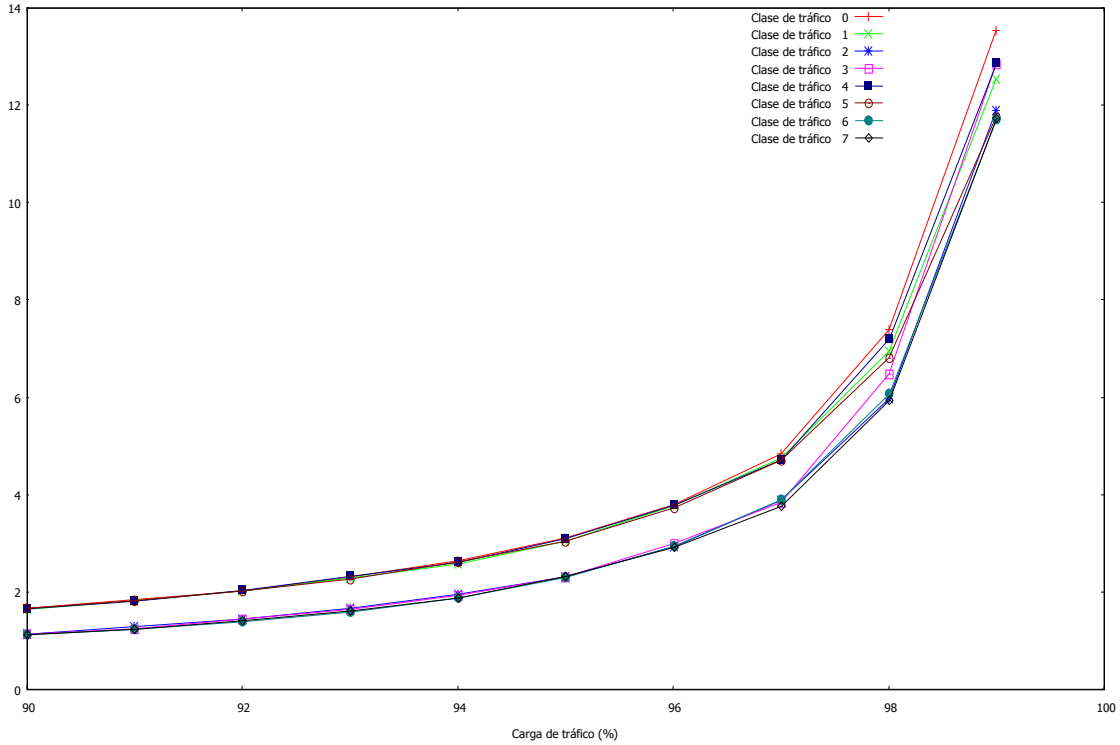

Figura 5.11: Retardo medio de encolado para el caso 3

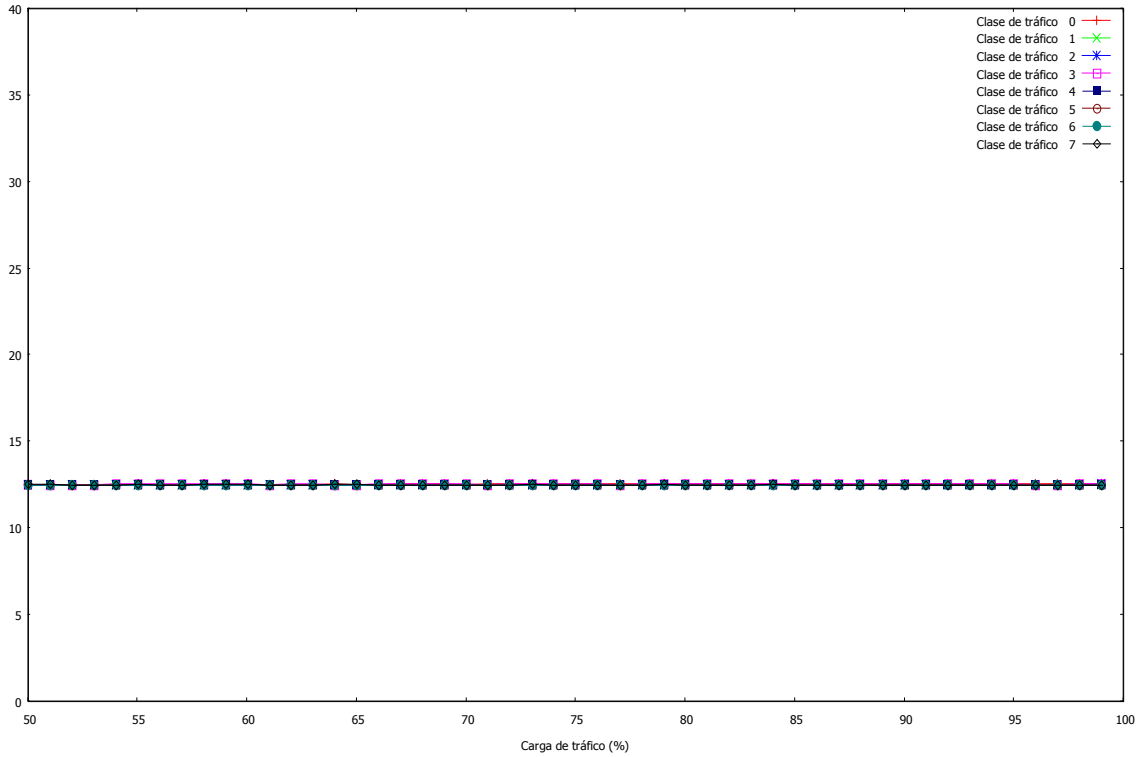

Figura 5.12: Ancho de banda asignado a cada clase para el caso 3

5.5.4 Caso 4: Control del retardo y del ancho de banda.

El objetivo de este supuesto es comprobar el efecto tanto de las prioridades como del ancho de banda en el retardo de los paquetes de una clase. En este caso se le asigna máxima

prioridad a las clases con mayor ancho de anda y mínima prioridad a las clases con menor ancho de banda. Como era de esperar las diferencias aumentan considerablemente, observándose gran desigualdad entre las clases más y las menos favorecidas.
La figura 5.14 muestra que el ancho de banda no sufre ninguna variación.

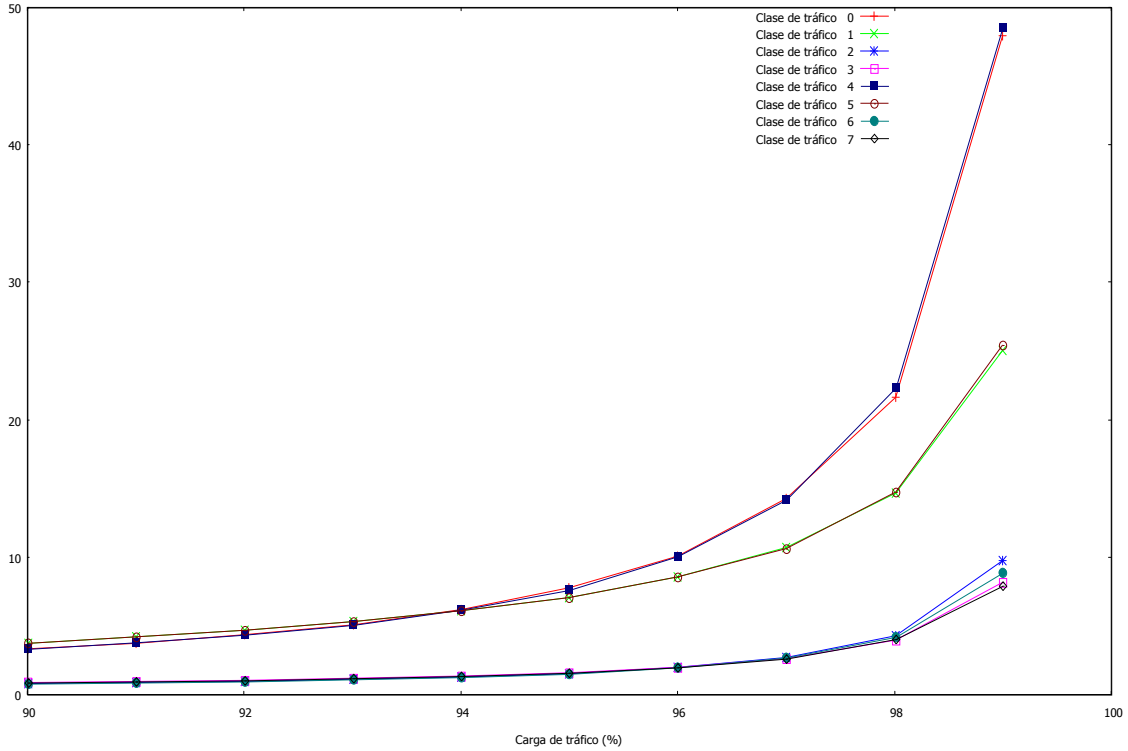

Figura 5.13: Retardo medio de encolado para el caso 4

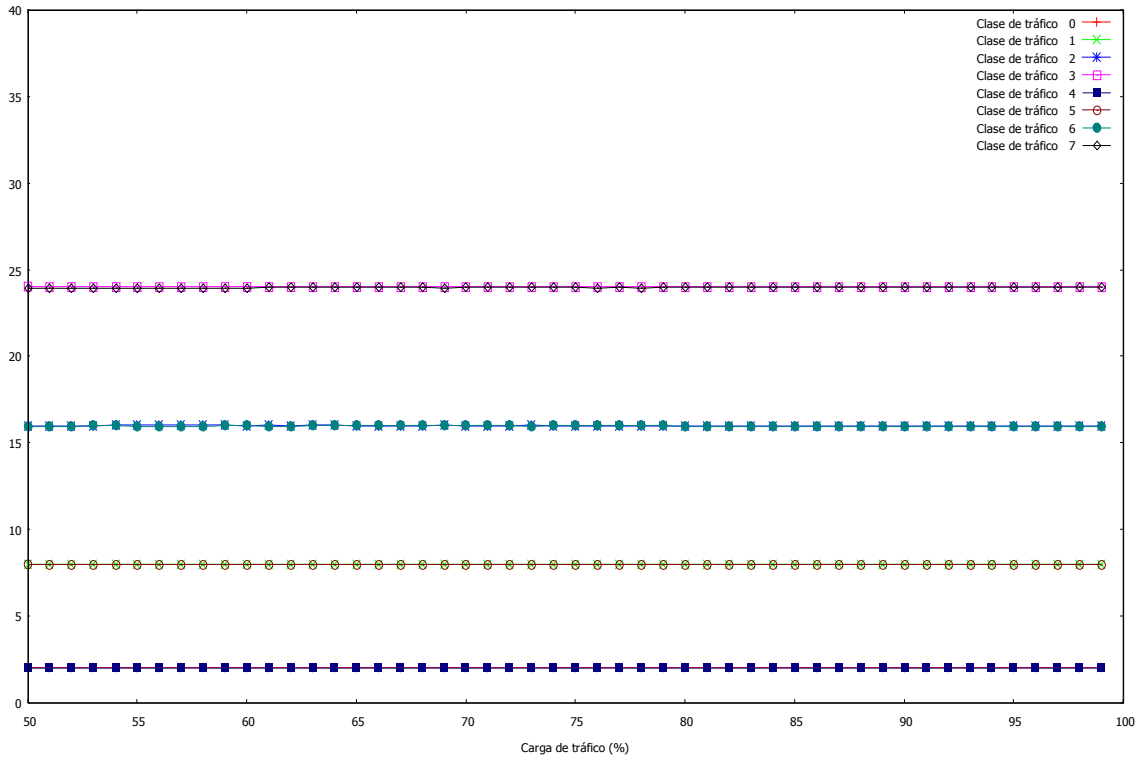

Figura 5.14: Ancho de banda asignado a cada clase para el caso 4

5.5.5 Caso 5: Control del retardo y del ancho de banda.

El objetivo de este supuesto es comprobar el comportamiento del retardo de una clase con prioridad mínima y ancho de banda máximo así como el extremo contrario (prioridad máxima y ancho de banda mínimo). Se recuerda que la asignación de anchos de banda es 2%, 8%, 16% y 24%. Teniendo esto, las clases con ancho de banda 2% y 8% se les ha asignado prioridad alta y a las clases con ancho de banda 16% y 24% se les asigna prioridad baja. En la figura 5.15 se puede observar que las clases más desfavorecidas han reducido su retardo (véase la figura 5.13) incrementándose ligeramente el retardo de las clases con mayor ancho de banda.

La figura 5.16 muestra que el ancho de banda no sufre ninguna variación.

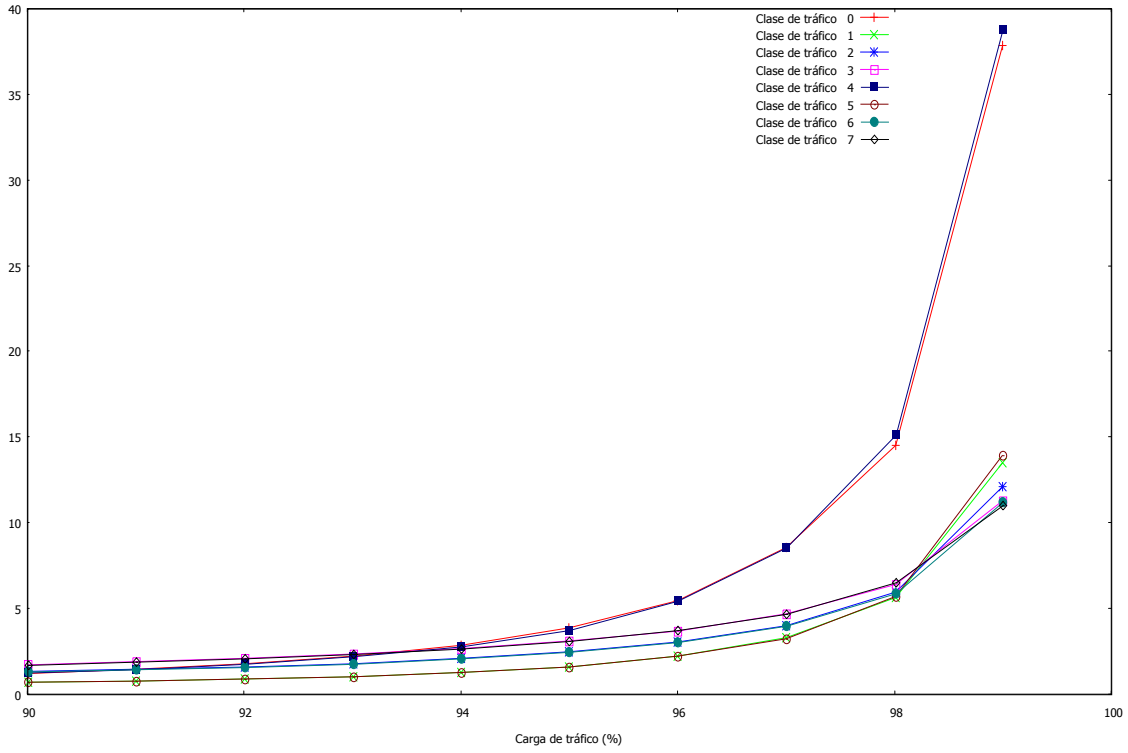

Figura 5.15: Retardo medio de encolado para el caso 5

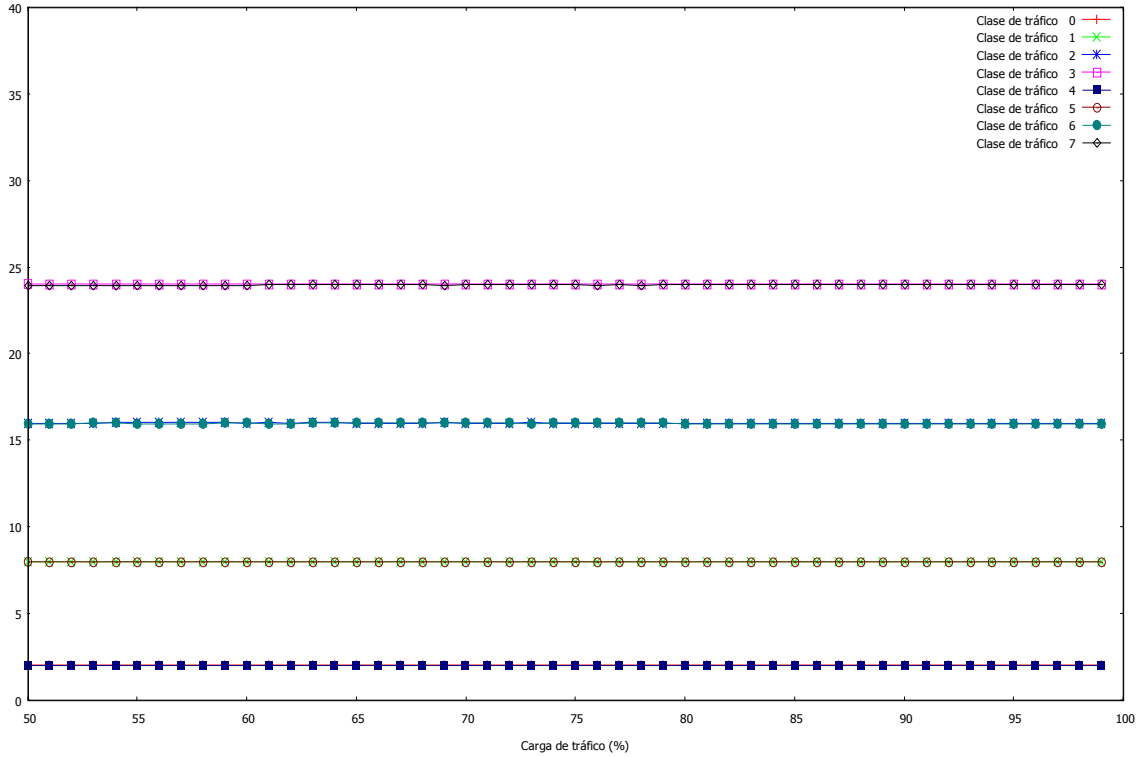

Figura 5.16: Ancho de banda asignado a cada clase para el caso 5

6. Conclusiones y líneas futuras

6.1 Conclusiones

A lo largo de este trabajo se han estudiado diversas arquitecturas de conmutadores, con multitud de algoritmos. Se ha pretendido abarcar lo máximo posible entre las arquitecturas existentes en la actualidad. La clasificación de los algoritmos presentados se ha realizado atendiendo a sus características intrínsecas y a la relación entre éstas y el rendimiento que permiten. Se ha insistido mucho, en la necesidad de escalabilidad, característica que va fuertemente ligada a la posibilidad de implementación hardware. También se ha hecho especial hincapié en la calidad de servicio. La necesidad de establecer características diferenciadoras entre los flujos de datos que direcciona un conmutador viene impuesta por la estructura interna de un satélite. De entre todos los algoritmos estudiados no se ha encontrado ninguno que satisfaga por completo las necesidades expuestas. Por ello ha sido necesario desarrollar un algoritmo propio capaz de garantizar las QoS necesarias, con alto rendimiento y con fácil escalabilidad.

La heterogeneidad de los sistemas embarcados obliga a diferenciar los flujos de datos que cada uno de ellos genera. Obviamente, no pueden tener la misma prioridad una orden enviada por el OBC (On Board Computer) al módulo de control de potencia (PCU) que una petición de telemetría recibida desde una estación de tierra. Así como tampoco se puede asignar el mismo ancho de banda la línea entre el OBC y los heaters (pequeñas resistencias colocadas por todo el satélite para controlar la temperatura de los instrumentos) que a la comunicación entre la unidad de captura de video y la unidad de estado sólido (disco duro de un satélite).

Se han presentado diversas soluciones comerciales ya implementadas en una plataforma física. El problema encontrado en todas ellas es que son soluciones parciales. Ninguna de ellas aprovecha abiertamente las características de los sistemas de rutado terrestres. Todas las arquitecturas comerciales encontradas están diseñadas según la filosofía de servicios integrados. Este hecho reduce en gran medida su eficiencia dado que la asignación estática de recursos provoca una mala gestión de los mismos.

Aunque no se ha dicho abiertamente, a lo largo del trabajo se han ido proponiendo y descartando diferentes arquitecturas en función de estas necesidades. La necesidad de escalabilidad (el objetivo final es implementar el algoritmo propuesto en un sistema hardware), las calidades de servicio, huir de las arquitecturas que necesitan aceleración (su coste en hardware es elevado y en un satélite cada gramo utilizado en la plataforma es un gramo que se puede usar en la carga de pago) o reducir al máximo la presencia de memorias (son elementos altamente sensibles a los SEU).

El resultado de esta evolución a lo largo de las diferentes arquitecturas es un algoritmo que se caracteriza por su eficiencia, flexibilidad y simplicidad. Este algoritmo ofrece diferentes calidades de servicio (las que se han observado como necesarias para un satélite)

garantizando siempre el envío de paquees, incluso de aquellos más desfavorecidos (menor prioridad y menor peso asignados).

Las gráficas presentadas, validan el modelo de simulación desarrollado para el presente trabajo, convierto este modelo, en una herramienta perfecta para la evaluación de algoritmos de planificación. El modelo de simulación de alto nivel, ha permitido generar todo tipo de estadísticas que facilitan el estudio del rendimiento de un conmutador.

Observando las gráficas expuestas se puede asegurar que el rendimiento del algoritmo presentado es en todo momento, mejor que el del VOQ, y en muchos casos cercano al rendimiento máximo, el perteneciente al OQ. Con estos datos se puede garantizar que el algoritmo propuesto tiene un alto rendimiento y que es capaz de gestionar flujos con diferentes QoS. Como se ha definido con anterioridad, estas características son imprescindibles para gestionar el tráfico de las comunicaciones internas de un satélite. Consecuentemente, podemos afirmar, que el modelo propuesto es un buen candidato para ser implementado en un router SpaceWire para el espacio.

6.2 Líneas futuras

Aunque en el presente trabajo se intentado hacer un análisis profundo del rendimiento del algoritmo de planificación son muchas las pruebas que se podrían realizar sobre éste para comprobar su comportamiento en entornos distintos.

Una de las posibles mejoras consistiría en comprobar el rendimiento con cargas de tráfico distintas a la uniforme. Sería interesante ver cómo reacciona el algoritmo ante tráfico de ráfagas o frente a cualquier otro que pueda aportar mayor información sobre el rendimiento del planificador.

También pudiera ser interesante modificar el algoritmo añadiéndole ciertas mejoras en cuanto a la calidad de servicio. Una posible mejora pudiera ser la posibilidad de filtrar paquetes recibidos, desechando aquellos que estén corruptos o simplemente obsoletos. Para ello, bastaría con poner colas en la entrada. Obviamente, esto incrementaría el retardo en la transmisión de los paquetes.

Por último, un trabajo altamente recomendable, pero de gran complejidad, sería implementar el algoritmo en un entorno hardware y tratar de validar los resultados aquí expuestos. Una plataforma idónea para este proceso pudiera ser una FPGA, dado que las características de estos dispositivos descritas durante el presente trabajo encajan a la perfección con las necesidades del router para el espacio.

Bibliografía

[ADF04] H.Z. Abidin, N.M. Din y N. Fisal, “Provisioning QOS using DiffServ with hierarchical scheduling,” IEEE Region 10 Conference TENCON 2004, vol. B2, pp. 597-600, noviembre 2004.

[AERO12] http://www.gaisler.com/doc/rt-spw-router.pdf Radiation-Tolerant 10x SpaceWire Router RT-SPW-ROUTER Data Sheet and User’s Manual, June 2012, Version 1.2

[AERO13]http://www.aeroflex.com/ams/pagesproduct/datasheets/spacewire/SpaceWire4-portRouter.pdf , Standard Products UT200SpW4RTR 4-Port SpaceWire Router Datasheet, April 8, 2013.

[AM95] R. Y. Awdeh y H. T. Mouftah, “Survey of ATM switch architectures,” Computer Networks and ISDN Systems, vol. 27, num. 12, pp. 1567-1613, noviembre 1995.

[AN89] M.M. Ali y H.T. Nguyen, “A neural network implementation of an input acces scheme in a high-speed packet switch,” Proceedings of IEEE Global Telecommunications Conference (GLOBECOM’89), vol. 2, pp. 1192-1196, noviembre 1989.

[AOS+93] T. E. Anderson, S.S. Owicki, J.B. Saxe y C.P. Thacker, “High-speed switch scheduling for local-area networks,” IEEE/ACM Transactions on Computer Systems (TOCS), vol. 11, num. 4, pp. 319-352, 1993.

[ATM13] http://www.atmel.com/Images/doc7796.pdf AT7910E SpW-10X SpaceWire Router DATASHEET, Rev:7796 GAERO 02/13

[BAE11] www.baesystems.com/download/BAES_052260/Space-Products--16--port-ASIC SpaceWire 16-port router radiation-hardened ASIC, BAE Systems2011

[BBC+98] S. Blake, D, Black, M. Carlson, E, Davies, Z. Wang y W. Weiss, “An Architecture for differentiated services,”, RFC 2475, Internet Engineering Task Force (IETF), diciembre 1998.

[BDE+04] H. Balakrishnan, S. Devadas, D. Ehlert y Arvind, “Rate guarantees and overload protection in input-queued switches,” Proceedings of IEEE Conference on Computer and Communications Societies (INFOCOM’04), vol. 4, pp. 2185-2195, marzo 2004.

[BZ97] J.C.R. Bennett y H. Zhang, “Hierarchical packet fair queuing algorithms,” IEEE/ACM Transactions on Networking, vol. 5, num. 5, pp. 675-689, octubre 1997.

[CF99] F.M. Chiussi y A. Francini, “Providing QoS guarantees in packet

switches," Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'99), vol. 2, pp. 1582-1590, diciembre 1999.

[CK04] N. Chrysos y M. Katevenis, "Multiple priorities in a twolane buffered crossbar," Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'04), vol. 2, pp. 1180-1186, noviembre-diciembre 2004.

[CLO01] H. J. Chao, C. H. Lam y E. Oki, Broadband packet switching Technology: A practical guide to ATM switches and IP routers, Wiley, 2001.

[CSIM] CSIM 19 Simulation Toolkit. Disponible en www.mesquite.com.

[CYR+04] K. Christensen, K. Yoshigoe, A. Roginsky y N. Gunther, "Performance of packet-to-cell segmentation schemes in input buffered packet switches," Proceedings of IEEE International Conference on Communications (ICC'04), vol. 2, pp. 1097-1102, junio 2004.

[DCS88] M. Devault, J. Cochennec y M. Servel, "The Prelude ATD experiment: assessments and future prospects," IEEE Journal on Selected Areas in Communications, vol. 6, num. 9, pp. 1528-1537, diciembre 1988.

[DKS89] A. Demers, S. Keshav y S. Shenker, "Analysis and simulation of a fair queueing algorithm," Symposium Proceedings on Communications Architectures & Protocols (SIGCOMM'89), pp. 1-12, septiembre 1989.

[DY02] V.L. Do y K.Y. Yun, "Packet latency optimization for VOQs in variable-length packet switches," Workshop on High Performance Switching and Routing (HPSR'02), pp. 77-82, mayo 2002.

[EC12] "SpaceWire – Links, nodes, routers and networks" ECSS-E-ST-50-12C, 31July2008

[EC13] "Interface and communication protocol for MIL-STD-1553B data bus onboard spacecraft", ECSS-E-ST-50-13C, 15 November 2008.

[EC53] "SpaceWire – CCSDS packet transfer protocol", ECSS-E-ST-50-53C, 5 February 2010.

[ES13] www.esa.int/Our_Activities/Space_Science/Gaia_overview
Última actualización 19 de Juno 2013

[GKS05] Y. Ganjali, A. Keshavarzian y D. Shah, "Cell Switching Versus Packet Switching in Input-Queued Switches," IEEE/ACM Transactions on Networking (TON'05), vol. 13, num. 4, pp. 782-789, agosto 2005.

[Gol91] S. J. Golestani, "A framing strategy for congestion management,"

IEEE Journal on Selected Areas in Communications, vol. 9, num. 7, pp. 1064-1077, septiembre 1991.

[HK88] M.G. Hluchyj y M.J. Karol, "Queueing in high-performance packet switching," IEEE Journal on Selected Areas in Communications, vol. 6, num. 9, pp. 1587-1597, diciembre 1988.

[IM01] S. Iyer y N. McKeown, "Techniques for Fast Shared Memory Switches," Stanford University, High Performance Networking Group, Technical Report TR01-HPNG-081501, 2001.

[JJACTEL] J. J. Wang, Brian Cronquist, John McCollum Actel Corporation, Rich Katz, Igor Kleyner (OSC) - NASA/GSFC Rocky Koga – Aerospace "Single Event Effects of a FLASH based FPGA"

[KKK90] C.R. Kalmanek, H. Kanakia y S. Keshav, "Rate controlled servers for very high-speed networks," Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'90), vol. 1, pp. 12-20, diciembre 1990.

[KHM87] M.J. Karol, M.G. Hluchyj y S.P. Morgan, "Input versus output queueing in a space division switch," IEEE Transactions on Communications, vol. COM-35, num. 12, pp. 1347-1356, diciembre 1987.

[KKL00] H. Kim, K. Kim y Y. Lee, "Hierarchical scheduling algorithm for QoS guarantee in MIQ switches," Electronics Letters, vol. 36, num. 18, pp. 1594-1595, agosto 2000.

[Kle75] L. Kleinrock, Queuing Systems Vol. 2 Computer Applications, Wiley Interscience, 1975.

[KP05] M. Katevenis y G. Passas, "Variable-size multipacket segments in buffered crossbar (CICQ) architectures," Proceedings of IEEE International Conference on Communications (ICC'05), vol. 2, pp. 999-1004, mayo 2005.

[KSC91] M. Katevenis, S. Sidiropoulos y C. Courcoubetis, "Weighted roundrobin cell multiplexing in a general-purpose ATM switch chip," IEEE Journal on Selected Areas in Communications, vol. 9, num. 8, pp. 1265-1279, octubre 1991.

[Lea90] C.-L. Lea, "Design and performance evaluation of unbuffered selfrouting networks for wideband packet switching," Proceedings of IEEE Conference on Computer and Communications Societies (INFOCOM' 90), vol. 1, pp. 148-156, junio 1990.

[LES09]http://lanzamientos.wordpress.com/2010/01/01/lanzamientos-espaciales-realizados-en-el-ano-2009/

[LK05b] T. H. Lee y Y.-C. Kuo, "Packet-based scheduling algorithm for

CIOQ switches with multiple traffic classes," Computer Communications (Elsevier), vol. 28, num. 12, pp. 1410-1415, enero 2005.

[Mak98] I.I. Makhamreh, "Throughput analysis of input-buffered ATM switch," IEE Proceedings of Communications, vol. 145, num. 1, pp. 15-18, febrero 1998.

[MBG+01] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi y F. Neri, "Packet scheduling in input-queued cell-based switches," Proceedings of IEEE Conference on Computer Communications (INFOCOM' 01), pp. 1085-1094, abril 2001.

[McK95] N. McKeown, "Scheduling algorithms for input-buffered cell switches", Ph. D. Thesis, Univerity of California at Berkeley, 1995.

[MMA+99] N. McKeown, A. Mekkittikul, V. Anantharam y J. Walrand, "Achieving 100% throughput in an input-queued switch," IEEE Transactions on Communications, vol. 47, num. 8, pp. 1260-1267, agosto 1999.

[MMW01] J. Mao, W.M. Moh y B. Wei, "PQWRR scheduling algorithm in supporting of DiffServ," Proceedings of IEEE International Conference on Communications (ICC 2001), vol. 3, pp. 679-684, junio 2001.

[MRS03] R. B. Magill, C. E. Rohrs y R. L. Stevenson, "Output-queued switch emulation by fabrics with limited memory," IEEE Journal on Selected Areas in Communications, vol. 21, num. 4, pp. 606-615, mayo 2003.

[MS01] S.-H. Moon y D. K. Sung, "High-performance variable-length packet scheduling algorithm for IP traffic," Proceedings of IEEE Global Telecommunications Conference (GLOBECOM'01), vol. 4, pp. 2666-2670, noviembre 2001.

[New92] P. Newman, "ATM technology for corporate networks," IEEE Communications
Magazine, vol. 30, num. 4, pp. 90-101, abril 1992.

[Nab00] M. Nabeshima, "Performance evaluation of a combined Input - and crosspoint- queued switch," IEICE Transactions on Communications, vol. E83-B, num. 3, pp. 737-741, marzo 2000.

[OMK+89] Y. Oie, M. Murata, K. Kubota y H. Miyahara, "Effect of speedup in nonblocking packet switch," IEEE International Conference on Communications (ICC'89), vol. 1, pp. 410-414, junio 1989.

[OMW06] S. O'Neill, A. Marshall y R. Woods, "Providing Input-Output Throughput Guarantees in a Buffered Crossbar Switch," Proceedings of 11th IEEE Symposium on Computers and Communications (ISCC 2006), pp. 725-730, junio 2006.

[PG93] A. K. Parekh y R. G. Gallager, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case,” IEEE/ACM Transactions on Networking, vol. 1, num. 3, pp. 344-357, junio 1993.

[PJB08] The; Blount, P. J. “ITAR Treaty and Its Implications for U.S. Space Exploration Policy and the Commercial Space Industry”, 73 J. Air L. & Com. 705 (2008)

[QLY+06] H. Qiu, Y. Li, P. Yi y J. Wu, “PIFO Output Queued Switch Emulation by a One-cell-Crosspoint Buffered Crossbar Switch,” Proceedings of International Conference on Communications, Circuits and Systems, vol. 3, pp. 1767-1771, junio 2006.

[RCG94] R. Rooholamini, V. Cherkassky y M. Garver, “Finding the right ATM switch for the market,” Computer, vol. 27, num. 4, pp. 16-28, abril 1994.

[SEN5]https://directory.eoportal.org/web/eoportal/satellite-missions/c-missions/copernicus-sentinel-5p Herbert J. Kramer ”Observation of the Earth and Its Environment: Survey of Missions and Sensors” 4th edition in 2002

[SMT98] D. Saha, S. Mukherjee y S. Tripathi, “Carry-over round robin: a simple cell scheduling mechanism for ATM networks,” IEEE/ACM Transactions on Networking (TON’98), vol. 6, num. 6, pp. 779-796, diciembre 1998.

[SPO6]https://directory.eoportal.org/web/eoportal/satellite-missions/s/spot-6-7 Herbert J. Kramer ”Observation of the Earth and Its Environment: Survey of Missions and Sensors” 4th edition in 2002

[SPS99] R. Schoenen, G. Post y G. Sander, “Prioritized arbitration for input queued switches with 100% throughput,” Proceedings of IEEE ATM Workshop, pp. 253-258, mayo 1999.

[SV96] M. Shreedhar y G. Varghese, “Efficient fair queuing using deficit round-robin,” IEEE/ACM Transactions on Networking, vol. 4, num. 3, pp. 375-385, Junio 1996.

[SZ98] I. Stoica y H. Zhang, “Exact emulation of an output queueing switch by a combined input output queueing switch,” Sixth International Workshop on Quality of Service (IWQoS’98), pp. 218-224, mayo 1998.

[TEA+03] F. Tobajas, R. Esper-Chain, V. de Armas, J.F. Lopez y R. Sarmiento, “Cell scheduling for VOQ switches with different strict priority levels,” Electronics Letters, vol. 39, num. 6, pp. 580-581, marzo 2003.

[TC94] Z. Tao y S. Cheng, “A new way to share buffer-grouped input queueing in ATM switching,” Proceedings of IEEE Global Telecommunications Conference (GLOBECOM’94), vol. 1, pp. 475-479,

noviembre-diciembre 1994.

[Tob90] F. Tobagi, “Fast packet switch architectures for broadband integrated services digital networks,” Proceedings of IEEE, vol. 78, num. 1, pp. 133-167, enero 1990.

[Tse05] E. S. H. Tse, “Switch fabric design for high performance IP routers: A survey,” Journal of Systems Architecture, vol. 51, num. 10-11, pp. 571-601, octubre-noviembre 2005.

[Wan01] Z. Wang, Internet QoS Architectures and Mechanism for Quality of Service, Morgan Kaufmann Publishers, 2001.

[CG02] H. J. Chao y X. Guo, Quality of Service Control in High-Speed Networks, John Wiley & Sons, 2002.

[WH07] F. Wang y M. Hamdi, “iPIFO: A Network Memory Architecture for QoS Routers,”Workshop on High Performance Switching and Routing (HPSR’07), pp. 1-5, junio 2007.

[WL02] M. Y. L. Wong y C. K. Li, “Low computational complexity weighted queuing using weighted deficit round robin,” Proceedings of 20th IASTED International Conference-Applied Informatics, febrero 2002.

[WWL05] C.-C. Wu, H.-M. Wu y W. Lin, “Efficient and fair multi-level packet scheduling for differentiated services,” Proceedings of the Seventh IEEE International Symposium on Multimedia (ISM’05), 7 pp., diciembre 2005.

[WWM+06] C.-C. Wu, H.-M. Wu, C. Moh y W. Lin, “A Hierarchical Packet Scheduler for Differentiated Services in High-Speed Networks,” Proceedings of the fifth IEEE/ACIS International Conference on Computer and Information Science (ICIS-COMSAR’06), pp. 63-68, julio 2006.

[YC01] K. Yoshigoe y K.J. Christensen, “A parallel-polled virtual output queued switch with a buffered crossbar,” Proceedings of IEEE Workshop on High Performance Switching and Routing, pp. 271-275, mayo 2001.

[YLZ03] M. Yang, E. Lu y S.Q. Zheng, “Scheduling with dynamic bandwidth allocation for DiffServ classes,” Proceedings of 12th International Conference on Computer Communications and Networks (ICCCN 2003), pp. 319-324, octubre 2003.

[YNO+02] K. Yamakoshi, K. Nakai, E. Oki y N. Yamanaka, “Dynamic deficit round-robin scheduling scheme for variable-length packets,” Electronics Letters, vol. 38, num. 3, pp. 148-149, enero 2002.

[YQW06] P. Yi, H. Qiu y B. Wang, “Implementing priority scheduling in a combined input-crosspoint-output queued switch,” Proceedings of 20th International Conference on Advanced Information Networking and Applications (AINA 2006), vol. 2, 5 pp., abril 2006.

[YWL+04] M. Yang, H. Wang, E. Lu y S.Q. Zheng, “Hierarchical scheduling for DiffServ classes,” Proceedings of IEEE Global Telecommunications Conference (GLOBECOM’04), vol. 2, pp. 707-712, noviembrediciembre 2004.

[ZH07] Y. Zhang y P.G. Harrison, “Performance of a Priority-Weighted Round Robin Mechanism for Differentiated Service Networks,” Proceedings of 16th International Conference on Computer Communications and Networks (ICCCN 2007), pp. 1198-1203, agosto 2007.

[Zha90] L. Zhang, “Virtual clock: a new traffic control algorithm for packet switchind networks,” ACM SIGCOMM Computer Communication Review, vol. 20, num. 4, pp 19-29, septiembre 1990.

[Zha95] H. Zhang, “Service Disciplines for Guaranteed Performance Service in Packet-switching Networks,” Proceeding of the IEEE, vol. 83, num. 10, octubre 1995.